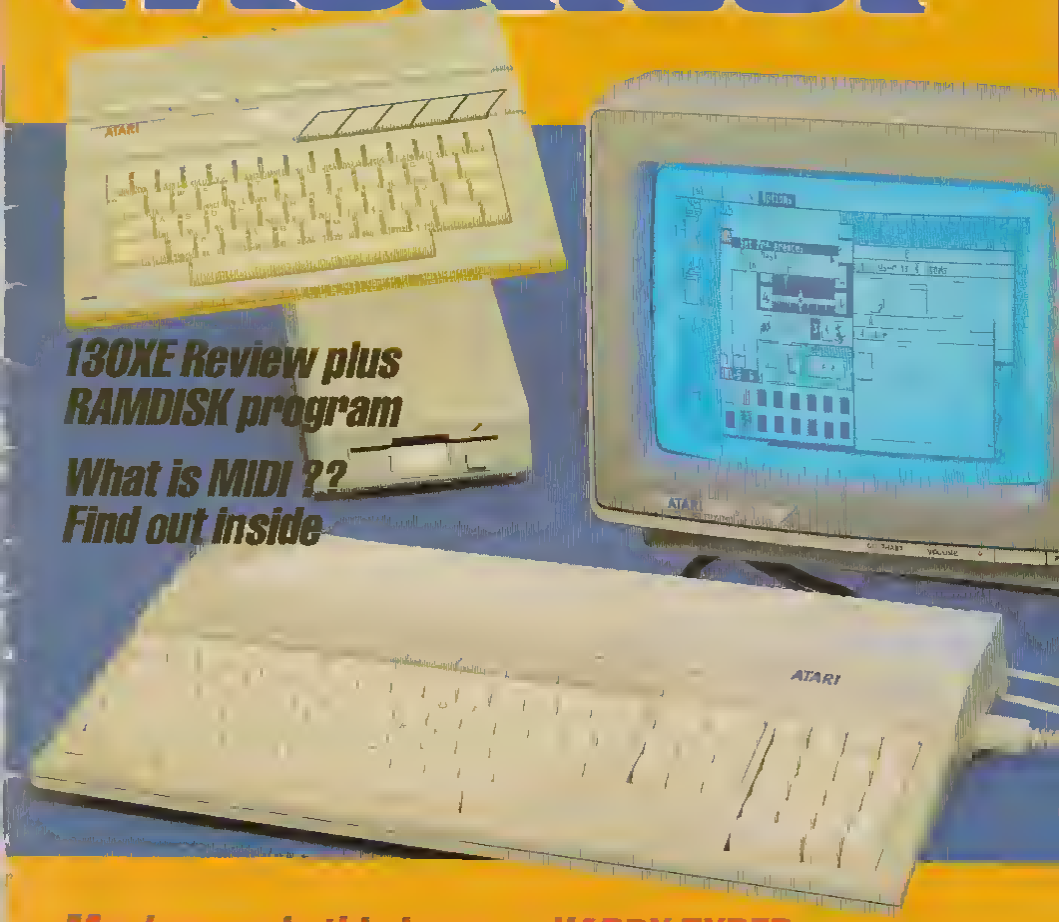
The U.K. ATARI Computer Owners Club Issue 9 Price £1.00

Independent User Group

Monitor



130XE Review plus RAMDISK program

What is MIDI ?? Find out inside

Much more in this issue including:
Fast Fill Routine
KEYO typing checker
Binary loads from Basic

HAPPY TYPER
Reviewed: TopDOS and Homeword
Profile on Lea Valley Club
Go FORTH

AVAILABLE FROM W H SMITHS, BOOTS, WOOLWORTHS AND MOST COMPUTER RETAILERS OR FROM LLAMASOFT 49 MOUNT PLEASANT, TADLEY, HANTS (TEL. 07356 4478) SAE FOR CATALOGUE & NEWSLETTER

'THE NATURE OF THE BEAST!

EXCITING TIMES!

Things are really moving on the Atari front these days. Since the new machines were announced all the national magazines have reviewed the 130XE and raved over the 520ST. It's as if they have suddenly realised what they have missed. If only they had asked us we could have told them years ago. I just feel sorry for all those people who have bought inferior products. At first they were sceptical that the new machines even existed, but now that the 130XE and the 520ST (in limited numbers at present) are here, they are getting more and more interested. Lets hope they continue as they have started and support the machines as much as they deserve.

It is reported that over one hundred development systems have been sold in the U.K. and that many well known software houses are busy developing (converting?) programs for the 520ST. Having the good fortune to see a 520ST system for myself recently, I can report that it lived up to all my expectations. GEM was very impressive, the mouse was easy to operate, the drives were fast and silent, the monitor was sharp and clear, the keyboard was finger sensitive and a joy to type on. If the 520ST package consisting of a computer/terminal, monochrome monitor, 3 1/2 inch disk drive, a mouse, GEM, BOS (business operating system), TOS, GEMpaint and GEMwrite, at £749 is not an all time winner, then I don't know what will be!

Atari DOS 2.5 is now being passed around. It contains extra files (not part of the DOS) to convert DOS 3 files to DOS 2.5 or DOS 2 format, unerase files, create an AUTORUN.SYS, verify disks and change drive numbers. Another file on the disk gives a RAMDISK for use with the 130XE. A mini manual is also supplied (supplied as a file on the U.K. version) which gives details on how to operate DOS 2.5, note that a full blown manual will be available later, probably costing around £14.

Atari User magazine is now well under way and Database has thanked the club for passing on the names and addresses of all the people who wished to subscribe.

Most of you have returned your membership questionaire and we are in the process of compiling the results. We wish to thank you all for your comments and we have already acted on some of the points raised. In this issue is our own typing checker program called KEYO which we think is a bit better than some others which are around at the moment. Also from this issue on you will be able to purchase a disk with all of the main programs on.

Finally, there are rumours that the next machine to be launched here will be the 256K model in the ST range, the 260STD (with built in disk drive). It will be interesting to see if this is true. One last point, we would like to credit some of the pictures in the 'Power without the Price' article in issue 8 to Analog magazine, our thanks to Michael J. Deschenes for writing to us to inform us of our omission.

CREDITS

Editor	Roy Smith
Art Editor	Peter Blackmore
Technical Editor	Ron Levy
Technical Editor	Keith Mayhew
Adventure Editor	Steve Hiller

CONTENTS

SUBSCRIPTION RATES

U.K. and Ireland	£4.00
Europe ..	£7.00
Outside Europe (Surface)	£7.00
Outside Europe (Airmail)	£10.50

The above prices are in English Pounds Sterling and include postage and packing.

A subscription/membership fee to join the U.K. Atari Computer Owners Club is just £4.00 for four issues of the club magazine. All cheques/postal orders are to be made payable to the 'U.K. Atari Computer Owners Club'. Overseas membership is also available at slightly higher rates. Overseas members who use the Library service should include enough extra monies to cover return postage.

ADVERTISEMENTS

Please note that the club cannot be held legally responsible for claims made by advertisers.

COVER: Photographs of the 520ST and the 130XE supplied by Silica Shop Ltd.
CLUB ADDRESS: The U.K. Atari Computer Owners Club, P.O. Box 3, Rayleigh, Essex.

130XE EXPLORED!

by Ron Levy

Welcome to this review of the first of Jack Tramiels new micro's – The 130XE

The 130XE is in essence, an 800XL (which was just a re-packaging of the trusty old 800/400 machines) but with a few layout changes and a VERY exciting addition in the form of an extra 64K of RAM!

The operating system and BASIC programming language of the 130XE is identical to the XL series so in this review, bearing in mind that the vast majority of our readers are already Atari computer owners, I shall restrict my comments to the features which make the 130XE unique within the Atari range. These are

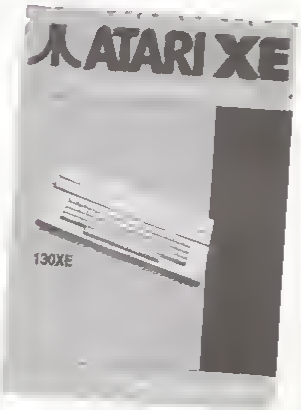
a) The Manual
b) The Physical Layout
c) The Extra 64K of Memory

THE MANUAL

Sadly, the manuals previously supplied with Atari computers have been shamefully inadequate. If you have bought an 800XL or 600XL you will have discovered that although you are given a nice glossy 61 page booklet, only four pages (7 sides) are of any actual relevance. The rest of the book is merely a set of the same 4 pages translated into five other languages! This is hopeless even for an experienced programmer. The novice doesn't stand a chance!

The new Atari regime, however has come to realise that a good machine is useless without a good manual, and that the vast majority of home computer buyers are novices. Thankfully the 130XE is supplied with a neat (8.5 * 5.5 inches) spiral bound handbook, an incredible 132 pages long and all in ENGLISH! At least one third of the manual is dedicated to teaching the elementary aspects of programming in BASIC to the absolute novice, and this it does very well indeed. It starts by showing how to print simple words to the screen and progresses at a gentle but thorough pace through to FOR/NEXT loops, mathematical calculations and IF/THEN comparisons.

The next section deals with sound and graphics, sound generation is dealt with fairly concisely – the book is adequate, but



not outstanding in this section. It is in its description of graphics, that the manual really shows signs of weakness. Although the modes 0 to 7 are dealt with in a very clear and concise way, and once again I think the beginner will find this very helpful, modes 8 to 15 are not even mentioned.

This I feel is a great shame, since these are probably the most interesting and versatile of the graphics modes available to the programmer. Modes 9 to 11 for example, have never really been explained in previous Atari manuals and yet they have been available since the days of the 400/800 machines. Their most interesting facet is the ability to produce realistic camera pictures with the use of a suitable video interface. Also amongst modes 12 to 15, there are some which are excellent for medium resolution 'painted' pictures (there is the special mode used by MICRO-PAINTER and other good 'painting' utilities), and these could do with explaining.

VERDICT – Although largely a vast improvement on previous efforts, this manual seems to stop abruptly halfway through the graphics section! It is very sad that Atari computers have the most advanced and varied graphics available on a home computer, and yet Atari still don't even tell owners about it in the manuals. Still, it is definitely a step in the right direction.

THE PHYSICAL LAYOUT

First impression is of the machines compactness. The reduction in width has been achieved by moving the function and reset keys to above the keyboard. Their diagonal shape has more to do with artistic appeal than practical useability, but nonetheless they do enhance the machines visual appeal, and the 130XE is certainly the smartest looking machine produced by Atari. I don't know whether it was intentional, but I found the ridge running across the machine (just above the logo) was very handy for laying pens in while programming!

There were, however, two aspects of the new machines case which I found rather irritating, the first of these being the location of the RESET key. This is placed just above the DELETE and BREAK keys, with an exposed edge facing them. Several times I have accidentally touched this key (it also has a very light spring), and since I use a disk drive, the computer cold-started itself with the inevitable loss of my program – most frustrating!

The other problem is in the characters printed on the keys of the main keyboard. They have been printed in a rather light grey colour, and consequently the finer punctuation keys require very good lighting conditions to locate them reasonably easily. I also thought that some of the punctuation marks were needlessly small. Indeed, I found the comma and full stop virtually impossible to differentiate without close examination, and this applies equally to the colon and semi-colon. Larger, bolder, black lettering would, I feel, have made the keyboard much nicer to use. As far as the keys themselves are concerned, I must say that I found them excellent. They have a beautifully light, rapid feel about them, I found that I could

type much faster on the 130XE than on my old 800 – something which I attribute to their relatively short travel and 'bouncy' feel.

A look inside the machine revealed the usual well screened, high quality circuit board which we have come to expect of Atari. Something I did find a little disappointing was that all the IC's were soldered directly into the PCB, thus making servicing much more difficult. I suspect that this has probably been done to help reduce costs, increase reliability, and cut down on inter-component interference, but it will be interesting to see how Atari deal with servicing, particularly on out of warranty machines when the time comes. It is interesting to notice that there are very few small scale logic IC's (3 TTL and 3 CMOS chips), an example of the trend towards large scale integration (LSI) IC technology, packing as many functions as possible into as few IC's as possible, hence the cluster of large IC's. We all, of course, benefit from this with cheaper, more powerful machines.

VERDICT – Despite the few niggly points, this is certainly the neatest and most professional looking Atari computer to date.

run a BASIC program any larger than the 800XL, in fact the 130XE will operate in precisely the same way as the 800XL. this is why it is 100% compatible with existing XL Atari software. The additional 64K exists as 4 separate 16K blocks and the only way it can be accessed is by disabling a 16K block of normal memory and placing a chosen block of the extra memory in its place. You can do this by poking to memory location 54017. Once this is done however, you can not access the original 16K memory block until it is re-enabled. Diagram 1 shows how this happens a little more clearly.

It is also possible to have the main processor (6502 CPU) accessing the system memory as normal, but have the display processor (ANTIC) accessing the extra RAM, and vice versa. This is because the control register in location 54017 uses 2 bits to decide which type of memory the two processors will use, one for the 6502 and one for the ANTIC. Two more bits are used to indicate which block of memory is to be used if the extra memory is selected by either of these.

Notice where the extra RAM is overlaid: it is in the second 16K section (locations 16384 to 32767). This is the obvious choice since the 0 to 16383 section holds the valuable system information which, if it is unavailable briefly, or was to get corrupted, would cause the computer to lock up, or crash. The third 16K RAM section (32768 to 49151) holds the screen data, and 8K of this is often switched out to allow BASIC or cartridge software to operate, so this is out of bounds. The top 16K section (49152 to 65535) is where the operating system lives in ROM, so it is also out of bounds. This only leaves the second 16K section, locations 16384 to 32767, to use as a 'window' to the extra RAM. This area is free when you switch on, but as your BASIC program grows, it will eventually reach and cover it. The problem then is that when you attempt to enable and access the extra RAM using BASIC commands (e.g. PEEK and POKE) you might actually disable your programmes memory space and crash BASIC!

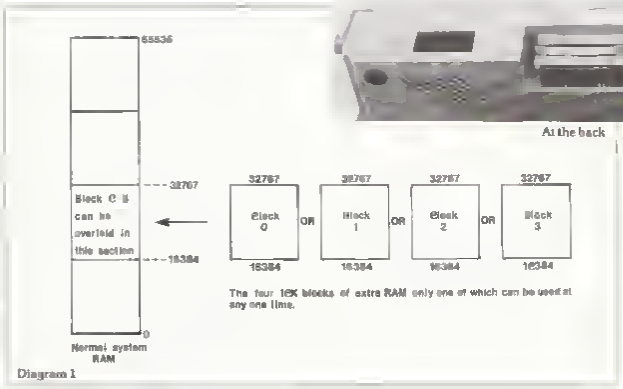


Diagram 1


At the back

THE EXTRA RAM

Now for the part which really does make the 130XE different to its predecessors. The 130XE is supplied with a total of 128K of random access memory (RAM), twice as much as the 800XL. Marvellous, I hear you say. But if you plug your machine in, switch on and type – PRINT FRE(0)

It responds with 37902, the same as it did with the 800XL (64K less the amount benked out to allow BASIC and the operating system to exist). Here is the catch, the additional 64K is not there to use as normal memory!

The problem is that the 6502 CPU, like all 8 bit processors with 16 bit address lines, can only 'talk' to up to 64K of continuous memory (ROM or RAM), in use.

So, where then is this extra 64K lurking in the 130XE? Firstly, you must realise that the 130XE will not be able to

This problem of conflict of memory would, at first, seem to prevent the use of the extra RAM by a BASIC program for data storage, but this need not necessarily be so. There are two ways around the problem –

a) Ensure that the BASIC routine which accesses the extra RAM lives outside the 'window' section. You can find the approximate position in RAM of a program line by including the following command on the same line – PRINT ADR("")

This seemingly pointless command will in fact print the address of the non-existent string in the quotes! Consequently you will know the approximate location (to within 4 or 5 bytes) of the program line.

b) Use a machine code subroutine located outside the second 16K section of memory to perform the access to the extra RAM. This is the fastest, neatest and most reliable method, but unfortunately not everyone knows how to write in machine code. For those who do, Atari BASIC provides an excellent method of interfacing machine code routines to a BASIC program, the USR function.

An application which the extra RAM is suited for is a RAMDISK. A RAMDISK is a large block of memory with a piece of machine code software which simulates a disk drive, so that programs which regularly access a file at random positions can be fooled into thinking the memory is a disk drive. This increases the speed of such a program enormously, as well as reducing drive wear.

Other applications which could benefit from having the extra memory are word processors, program text editors or database type programs. Atari have stated that any software packages produced for the 130XE will be made compatible with the older Atari computers, but perhaps with some features lost. It will certainly be interesting to see whether they stick to this.

VERDICT – The extra 64K is a superb feature which should ensure the survival of the Atari machines. It opens up the wey for many exciting possibilities such as a RAMDISK, large database programs, perhaps even a decent 'spelling checker' type word processor. Adventure games that use graphics can be vastly improved avoiding the need to load each screen separately from disk.

When you take all the possible future benefits into consideration, and the fact that by upgrading you will not loose out to any kind of incompatability with the software you already have, finding the measley £170 to buy one is a temptation not easy to overcome. It will be an investment worth every penny!

I would like to thank Silica Shop for sending me the 130XE for review so soon after they arrived in the U.K. Its nice to know that someone out there is supporting the Atari computers with enthusiasm.

Binary Loads From BASIC

by Steve Hillen

There have always been a number of problems in loading a binary load file from BASIC These problems stem mainly from the fact that another DOS file must be loaded itself in order to load a binary file for you This file will either overwrite your BASIC program or force you to wait for ages while a section of memory is saved out under a MEM SAV file. Another problem is that the DOS loader will protect itself and not allow any loads over its region of memory, a region which is normally user space

There are 3 answers to the problem Firstly, get another DOS e.g. OS/A+ which has a resident binary load option Secondly, make your binary load file into an AUTORUN SYS file, so it is loaded on boot up bypassing the second DOS file These two solutions are either expensive or inconvenient, so perhaps the compromise solution is to have a short routine to perform the load for you

Before the listing explanation how ever, a quick reminder on the structure of a binary file

STRUCTURE

It is basically a memory dump onto disk (or cassette) with a special 6 header bytes which determine where to load the file, and how much there is to load Referring to Figure 1 the first two bytes are a mandatory 255,255 which specify that it is a binary file The next two bytes are the lo hi address at which to start the load The 5th and 6th bytes are the lo hi address at which the load will end. Following the first 6 bytes is the data which is to be loaded between the two addresses

A binary load file does not stop there though Depending on how it was created it could have a number of supplementary files appended to the end Such a file is known as a compound file. The only difference between these extra sections and the first, is that the first two 255's may or may not be included The four address bytes and the specified amount of data will still be there, however

The listings below are for a BASIC binary load option, which occupies most of Page 6 in memory Once installed, you can call it with —

X=USR(1536,ADR("D:FILENAME.EXT"))

Where you supply the filename you want to load. At the moment the routine will not load any program between page 6 and page 31, in order to protect itself and the resident DOS The upper and lower bounds can be altered by poking 1605 and 1609 respectively with the new bounds

You will also notice that the program will print the memory ranges loaded in hex, and there are two things to look out for If the file loads into $2E0-$2E1 then those two locations hold the RUN address for the file If it loads into $2E2 $2E3 then those two locations hold the INIT address for the file

Type in listing 1 and save it out before running it As it stands the program will read the data into page 6 then stop If you delete line 10 however the program will create a binary load file of itself which you can load once with DOS, then load any further binary files without the need for MEM.SAV etc

Listing 2 is the assembly code and is only shown for more advanced readers who may wish to see how it works

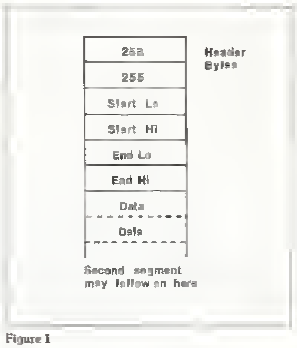


Figure 1

NOTE In this program, anything which is underlined, should be entered in 'INVERSE'

```
EI 1 REM ********************************
NH 2 REM
JG 3 REM BINARY LOAD FROM BASIC
NJ 4 REM
TF 5 REM BY STEVE HILLEN
NL 6 REM
YH 7 REM FOR MONITOR MAGAZINE
EQ 9 REM ********************************
GH 10 GOTO 31000
VO 100 REM *** Create binary load file. *
   **
EE 110 DIM IN$(1)
IN 120 ? "Ready disk with DOS and press <
   CR>":INPUT IN$
KG 130 TRAP 200
JF 140 OPEN #1,8,0,"D:BINRHYL.OBJ"
YH 150 PUT #1,255:PUT #1,255:PUT #1,0:PUT
    #1,6:PUT #1,244:PUT #1,6
IN 160 FOR A=0 TO 244:READ D:PUT #1,D:NEX
   T A
PG 170 CLOSE #1:? "Done.":END
DI 200 ? "Error ";PEEK(195):CLOSE #1:END
RG 31000 REM *** Load data onto page 6 **
   *
QH 31001 FOR R=0 TO 244:READ D:POKE 1536+
   R,D:NEXT R
MX 31002 REM *** Ready to use. ***
XO 31003 REM *** X=USR(1536,ADR("D:FILEMA
   NE.EXT")) ***
YO 31004 END
VQ 31010 DATA 104,216,162,16,169,3,157,66
   ,3,169,4,157,74,3,104,157,69,3,104,157
   ,68,3,32,86,228,48,116,32
TU 31011 DATA 123,6,201,255,208,109,32,12
   3,6,201,255,208,102,160,0,132,203,32,1
   23,6,164,203,153,208,0,200,192,4
FP 31012 DATA 144,241,32,152,6,165,208,15
   7,48,3,165,209,201,6,144,4,201,31,144,
   67,157,69,3,56,165,210,229,208
BR 31013 DATA 157,72,3,165,211,229,209,15
   7,73,3,254,72,3,208,3,254,73,3,32,131,
   6,32,123,6,133,208,32,123
LB 31014 DATA 6,201,255,240,188,133,209,1
   60,2,208,176,169,0,157,72,3,157,73,3,1
   69,7,157,66,3,32,86,228,16
GT 31015 DATA 10,104,104,169,12,157,66,3,
   32,86,228,96,160,0,165,209,32,208,6,16
   5,208,32,208,6,208,165,211,32
IZ 31016 DATA 208,6,165,210,32,208,6,162,
   0,169,9,141,66,3,142,73,3,169,127,141,
   72,3,169,6,141,69,3,169
VE 31017 DATA 235,141,68,3,32,86,228,162,
   16,96,162,1,72,74,74,74,74,9,48,201,58
   ,144,2,105,6,153,235,6
AM 31018 DATA 200,202,48,6,104,41,15,76,2
   13,6,96,70,70,70,70,45,70,70,70,70,155
```

Listing 1

Listing 2

```
00001 ;Binary load files from Basic
00002 ;Written on SynAssembler
00003 ;
00004 ;Called by
00005 ;X=USR(1536,ADR("D:FILENAME.EXT"))
00006 ;
00007           .LI OFF
00008           .OR $600    Page 6
00009           .TF "D:BINLOAD.OBJ"
00010 ;
00011 ;CIO Equates
00012 ;
00013 ICCMD     .EQ $342    Command byte.
00014 ICBAL     .EQ $344    Buffer address low.
00015 ICBAH     .EQ $345    Buffer address high.
00016 ICBLL     .EQ $348    Buffer length low.
00017 ICBLH     .EQ $349    Buffer length high.
00018 ICAX1     .EQ $34A    Auxiliary byte 1.
00019 ;
00020 ;CIO commands
00021 ;
00022 OPEN      .EQ 3       Open file.
00023 CLOSE     .EQ 12      Close file.
00024 GET       .EQ 7       Get bytes.
00025 PUT       .EQ 9       Put bytes.
00026 ;
00027 CIOV      .EQ $E456   CIO entry vector.
00028 ;
00029 ;My workspace.
00030 ;
00031 BUFFER    .EQ $D8     Four bytes space.
00032 TEMP      .EQ $C0     One byte space.
00033 ;
00034 ;
00035 START     PLA         Clear stack.
00036           CLD         Clear decimal.
00037           LDX #$10    Set to IOCB #1.
00038           LDA #OPEN   Open the file.
00039           STA ICCMD,X
00040           LDA #4      Direction is read only.
00041           STA ICAX1,X
00042           PLA         Filespec address
00043           STA ICBAH,X is passed on the
00044           PLA         stack and saved
00045           STA ICBAL,X into the buffer addr.
00046           JSR CIOV    Let CIO open file.
00047           BMI ENDED   If Y -ve then error.
00048 ;
00049           JSR GET1    Get one byte from file.
00050           CMP #$FF    Must be 255 for
00051           BNE ENDED   a binary load file.
00052           JSR GET1    Save for second byte.
00053           CMP #$FF
00054           BNE ENDEN
00055 ;
00056 AGAIN     LDY #0      This loads 4 bytes
00057 GET4      STY TEMP    from the file
00058           JSR GET1    and saves them in
00059           LDY TEMP    buffer- then 4 are
00060           STA BUFFER,Y the start and end
00061           INY         load addresses for
00062           CPY #4      the file.
00063           BCC GET4
00064 ;
00065           JSR PRINT   Print the start/end addr.
00066 ;
00067           LDA BUFFER  The start load goes
00068           STA ICBAL,X into the buffer addr.
00069           LDA BUFFER+1 Check that file will
00070           CMP #6      not overwrite
00071           BCC OK      above Page 6
00072           CMP #$1F    and below
00073           BCC ENDED   page $1F (protect DOS)
00074 OK        STA ICBAH,X
00075           SEC         Now subtract start
00076           LDA BUFFER+2 addr. from end
00077           SBC BUFFER  addr. and add 1
00078           STA ICBLL,X to get the length
00079           LDA BUFFER+3 of the file to load.
00080           SBC BUFFER+1
00081           STA ICBLH,X
00082           INC ICBLL,X
00083           BNE JUMP
00084           INC ICBLH,X
00085 JUMP      JSR PART2   Load the file.
00086 ;
00087           JSR GET1    Test for compound file.
00088           STA BUFFER  Save for later
00089           JSR GET1    Get another-if it's 255
00090           CMP #$FF    then need another 4 bytes.
00091           BEQ AGAIN   If not, then those 2
00092           STA BUFFER+1 were the start load addr.
00093           LDY #2      Specify two bytes only
00094           BNE GET4    and get them!
00095 ;
00096 GET1      LDA #0      If length set to 0,
00097           STA ICBLL,X one byte is returned
00098           STA ICBLH,X in A.
00099 PART2     LDA #GET    Set for get byte.
00100           STA ICCMD,X
00101           JSR CIOV    Get byte.
00102           BPL BACK    No error if Y +ve.
00103           PLA         Clear last RTS.
00104           PLA
00105 ;
00106 ENDED     LDA #CLOSE  Close the file.
00107           STA ICCMD,X
00108           JSR CIOV    and exit to Basic.
00109 BACK      RTS
00110 ;
00111 PRINT
00112           LDY #0      This section changes
00113           LDA BUFFER+1 the first 4 hex
00114           JSR HTOP    bytes in Buffer
00115           LDA BUFFER  to a printable 8 bytes.
00116           JSR HTOP
00117           INY
00118           LDA BUFFER+3
00119           JSR HTOP
00120           LDA BUFFER+2
00121           JSR HTOP
00122 ;
00123           LDX #0      Specify IOCB 0
00124           LDA #PUT    Put a record.
00125           STA ICCMD   Set buffer length.
00126           STX ICBLH
```

```
00127           LDA #$7F
00128           STA ICBLL
00129           LDA /MESSAGE Point to message to
00130           STA ICBAH    print to screen.
00131           LDA #MESSAGE
00132           STA ICBAL
00133           JSR CIOV     Let CIO print.
00134           LDX #$10     Restore IOCB 1
00135           RTS
00136 ;
00137 ;
00138 HTOP      LDX #1       Set Index.
00139           PHA          Save for later.
00140           LSR          Shift high nibble
00141           LSR          to low nibble.
00142           LSR
00143           LSR
00144 CORRECT   ORA #$30     Put in $0C11 offset.
00145           CMP #$3A     And correct for hex
00146           BCC NOTHEX   numbers.
00147           ADC #6
00148 NOTHEX    STA MESSAGE,Y Store.
00149           INY
00150           DEX
00151           BMI RET      Both done.
00152           PLA          Restore A
00153           AND #$0F     Get low nibble.
00154           JMP CORRECT and change to ASCII.
00155 RET       RTS
00156 ;
00157 MESSAGE   .AS "FFFF-FFFF"
00158           .HS 9B
```

CROSSWORD COMPETITION RESULT

We had an overwhelming response to the crossword competition and it took a lot of work to sort out the correct entries and put them into the proverbial hat ready for picking the winner Surprisingly there was only seven correct entries so we probably made the clues a bit too difficult. Next time we will make them a bit easier!

Our congratulations to Liz Ahmedzai from Chatham in Kent for being one of the seven and for having the good fortune to have her name pulled from the hat. We hope the £10 came in handy Liz! Also our condolences to the six who got it right but get nothing for their efforts

To put all of you who are curious about the answers out of your misery, here they are

Answers ACROSS 3)Jack Tramiel, 6)Interface 9)ROM 11)Byte 12)List, 13)Binary 15)Page 17)Command, 19)Tiny, 20)Sector, 22)Terminal, 24)Disk Drive, 26)Mode, 30)Decimal, 31)Recursive 34)Via Tape 35)Language 37)Integer, 38)Keyboard, 39)Open

Answers DOWN 1)Antic, 2)Zero, 4)Error, 5)Bit, 7)Cassette Recorder 8)Structured, 10)Manual, 14)Joystick, 16)System, 18)String, 21)DIM, 23)Atari, 25)Array 27)Next 28)Nibble, 29)New 30)Data 32)Even, 33)Bug 35)Loop, 36)Game

MONITOR ON DISK

Like the look of a program but can't find time to key it in? You've asked the wife three times to do it for you whilst you're out at work, and she still hasn't. Or maybe you have typed it in but it won't run. Then why not take all the effort out of it and send for the MONITOR DISK. All the main programs in each issue of MONITOR are now available pre recorded on disk for you. They cost £4 95 which includes postage and packing, send a cheque/postal order made payable to the 'U K Atari Computer Owners Club to Monitor Magazine P O Box 3 Rayleigh Essex If you live in Europe add 50p if outside Europe add £1 00 Please allow 28 days for delivery

Monitor Disk 8
Includes Quickplot, a fast Graphics 8 Plot/Drawto handler Nightmare Reflections, an exceedingly frustrating adventure Matchbox, improve your concentration with this memory game Interrupts 5 demo programs showing various uses of interrupts

Monitor Disk 9
Includes Keyo, a new typing checker Multiboot Bootbase database program for Multiboot disks' Binload, binary loads from Basic. Happytyper automatic line numbering. Ramdisk, for use with the 130XE Fast Fill a speedy shape filling utility

STOP PRESS STOP PRESS

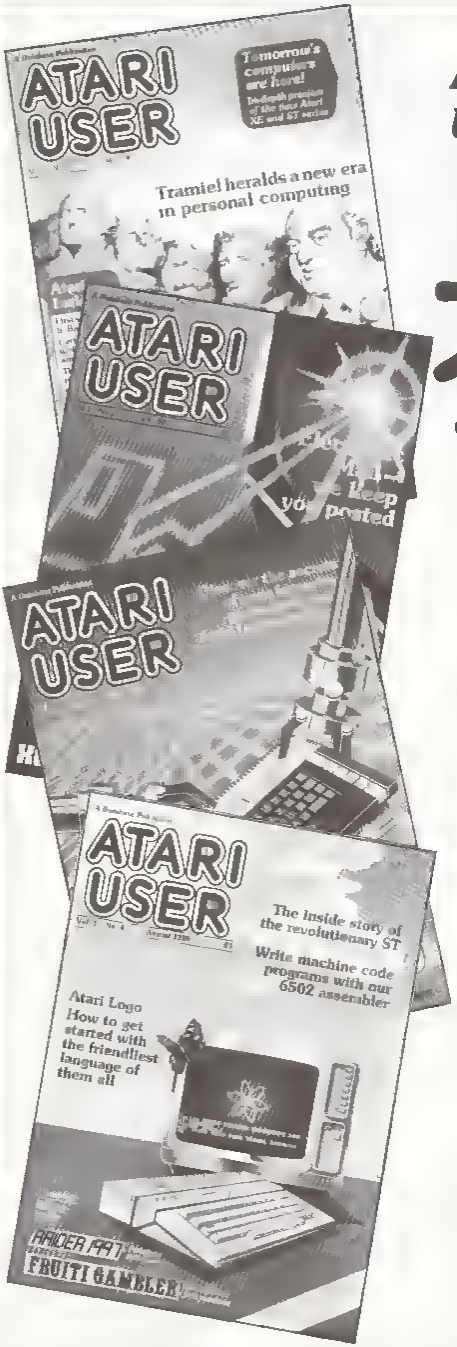
A new word processing package is soon to be released (August) by Precision Software for use on the XL/XE range It will be supplied on disk with a fully comprehensive manual which includes a training course The word processor is menu driven with single command selection plus user defined selection It also has a calculator, mail merge facilities, a dictionary of over 20 000 words (which is also user definable) and can be used as a spelling checker Various Printer set up files are included (yes, these are user definable too) Although it is menu driven, for more experienced users all commands can be inserted directly into the text. The package is being marketed under the 'Superscript' brand name and will be on display at the PCW show at Earls Court in September The price will probably be £69 95, which sounds a lot but after having seen a preview copy and being totally impressed with the quality of the program came to the conclusion that it will be a bargain!

A new Magical adventure from Level 9 called 'Red Moon' is to be released on 15th July It is cassette based text with graphics and will retail for £6 95 If it is as good as their others its worth looking out for

Discovered in amongst the pile of letters we receive at H Q was the first issue of a new newsletter for Atari fans It is pretty good, we liked the 'Atariman' cartoon strip best (wish we had thought of it!) Named 'GTIA Gazette' details are available from Mike Lynch, 24 Oakdene Road, Anfield, Liverpool, L4 2SR

On the subject of newsletters we have been sent a copy of 'Atari Magazine' which is published by Stichting Atari Gebruikers in Holland. It is a professionally printed newsletter full of tips and listings and although its all in Dutch and we could'nt read it, it was fun typing in the programs just to find out what they did. If you are interested, we are sure Casper Jansen would love to here from you (no, you don t have to write in Dutch, he speaks English) Write to Postbus 40181, 6504 AD Nijmegen, Holland.

There are very strong rumours that Atari Corporation have booked three massive stands at the forthcoming PCW show at Earls Court in September we have also heard that Silica Shop and Precision Software will be there, and its a possibility that Page 6 magazine will have a stand too If all this comes to pass it will be a show not to be missed by any Atari enthusiast.



A special offer for User Group members!

£3 OFF

each new subscription

Here's your chance to make sure of a regular copy of Atari User – the independent magazine that's become the premier source of information on the whole range of Atari microcomputers.

Each month brings you a choice selection of first rate programs, powerful utilities, easy to follow articles and invaluable hints and tips – plus full coverage of what's happening in the world of the Atari, with in-depth reviews of all the latest products.

If you want to know all there is to know about your Atari micro, Atari User is essential reading!

ORDER NOW!

Pleases send me the next 12 issues of Atari User at the specially reduced User Group members' price of £9 (normal price £12)
Please indicate method of payment

☐ Access/Mastercard/Eurocard

☐ Barclaycard/Visa

☐ Cheque/PO made payable to Database Publications Ltd

Name _____

Address _____

_____

Signed _____

Send to: *Atari User*, FREEPOST, Europa House, 68 Chester Road, Hazel Grove, Stockport SK7 5NY

(No stamp needed if posted in UK)

STARTING FROM BASICS

by Captain Hacker

Welcome to the second part of my column for the beginner. This is where I hope to alleviate the confusion experienced by the novice who wants to get a foothold into the world of programming. In this issue I will deal with the way elements of information are held in the computer. There are two types of information (or data) that can be manipulated in a program, the first is numeric values, and the second is letters, or words.

Numbers

In the last issue I told you a little about the PRINT command, and how to use it to print your name to the screen. Since I will now use this command it is important that you read and understand the article in the last issue (Back issues are available). Let us suppose that we want to print the number 2 onto the screen. This is very simple just type –

```
PRINT 2
```

Press RETURN on your computer and you will see that it obediently places a 2 on the next line, just as instructed! Suppose though, that you want to do something more useful with numbers, what can your computer do? Well in fact, your computer can perform any of the basic mathematical operations for which you might normally use a calculator, i.e. add, divide, subtract and multiply. It can also perform the kind of functions which you would normally find on a good scientific calculator, such as SIN, COS, LOG, etc.

If I decide that I want my machine to add two and four together, I would simply type –

```
PRINT 2+4
```

Try it, and you will see that it prints 6 on the next line. What about subtraction though? Type –

```
PRINT 4–2
```

This, of course, gives the answer 2. When adding or subtracting I use the same keys as on a calculator but for multiplying and dividing I have to use different symbols. For multiplying I use the ASTERISK, i.e. –

```
PRINT 4*2
```

With the result of 8 being printed.

For division I use the symbol which is refered to by programmers as a SLASH i.e. –

```
PRINT 4/2
```

This gives the value of 4 divided by 2. All these symbols are on the lower right hand side of your keyboard. Experiment with them for a while, and remember that they can be used in a combination, thus. –

```
PRINT 4*3/2.5+1
```

Numeric Variables

Let us suppose that I want to hold a numeric value in a program but I want to alter it in various ways as the program progresses. To do this I have to use a VARIABLE. You can imagine this as a pot, into which I may place a number. I can alter the value in this pot using any of the mathematical functions I experimented with earlier, and I can have as many 'Pots' as I wish but how do I refer to them, or tell them apart from each other? I just label each pot seperatley, just as if I were actually labelling different types of jam. I stick imaginary labels onto my imaginary 'Pots' I refer to the label as the VARIABLE NAME.

To create a number pot I need only refer to its name and the computer will automatically make a space for it in its memory. Type the following: –

```
A=1
```

Notice that when you press RETURN the computer does not print anything. All it has done is to create a number 'Pot', place the value 1 into it, and then labelled it with the letter 'A'. Many computers only permit VARIABLE NAMES of one character length (i.e. A,B,C,D etc) but fortunately the Atari allows names of almost any length. The only limitation is that you must NOT use words which are themselves commands such as PRINT or GOTO, (these are called reserved words). I can print the contents of a variable in the same way as I would print a literal number, e.g. –

```
PRINT A
```

In fact, I can treat number variables in exactly the same way as I would treat numbers themselves i.e. –

```
PRINT A+5
```

The answer will be 6 since the last operation I performed on the variable A was to force the number 1 into it. Experiment with the following command lines on your screen –

```
A=A+2
PRINT A
B=7
PRINT A+B
PRINT A*B
PRINT A/B
```

To show how to use a number variable in a program, here is a short program for you to type in –

```
10 A=1
20 PRINT A
30 A=A+1
40 GOTO 20
```

Type RUN after you have entered each line but remember that to stop the program you will have to press the BREAK key!

String Things

The second kind of information dealt with in a computer is called STRINGS. A string is the term used to describe a collection of letters grouped together, usually to form a word. Remember that in the last issue I used the PRINT command to print a neme to the screen, thus –

```
PRINT "JACK"
```

Well let us suppose that, rather than quote the name directly I would prefer to store the name JACK into one of our imaginary pots, (the same as I did with numbers) and give it a label. I can do this, but I somehow need to show the computer that I am dealing with a string rather than a number variable. This is done by putting a dollar sign after the variable name (or label). For example, whereas I use the name A for our number variable, I must

use A$ as the name for my string variable (the dollar sign is always refered to by programmers as STRING – i e B$ is read as 'B string')

It is here, however, that I encounter a slight problem Numbers are stored in the computer s memory in a specially coded form which requires a fixed amount of room, regardless of the value Strings are a little more awkward Because I might want my imaginary pot (or string variable) to hold either a short name or perhaps a whole sentence the amount of memory space needed to hold a string variable might vary enormously and this poses a problem for the computer What I need to do is to somehow tell the computer how much room to allow for my string variable This I do at the very beginning of my program, using the DIM command (short for DIMension) Suppose that I decide I want to use a string variable named A$ and that the longest I think it will be is ten characters I declare this with the following line –

```
10 DIM A$(10)
```

I can now use the variable A$ and place whatever characters I want into it. For example –

```
20 A$="JACK"
```

Type NEW, and enter lines 10 and 20 as above, then type RUN You have now stored the word JACK in the variable A$ Notice, however that although line 10 prepared A$ for up to ten characters you only used four of them This is an important point, since in line 10 you told the computer that you MIGHT want to have A$ up to 10 characters long but since you do not have to use all of the space you have reserved for it the computer also keeps track of how long A$ really is If you now type.–

```
PRINT A$
```

The name JACK has dutifully been stored in the variable labelled A$ and as you will see you can print its contents in the same way as a number variable Type LIST and you will again see the program lines you have entered Now type NEW and try LIST again You will find that your program no longer exists! This is the way to clear the computer's memory and tell it that you want to enter a different program. Type in the following. –

```
10 DIM A$(10) B$(10)
20 A$= JACK" B$="SMITH"
30 PRINT A$ PRINT B$
40 PRINT A$ B$
50 PRINT A$ B$
60 PRINT A$ – ",B$
```

Type RUN and you should see the following printed on your screen –

```
JACK
SMITH
JACK          SMITH
JACKSMITH
JACK  SMITH
```

Examine the program carefully Notice that I was able to DIMension A$ and B$ at the same time by separating them with a comma. Notice also that as on line 20 it is possible to have more than one separate command on one line number To do this you must separate the two commands with a COLON On line 20 the two commands are both PRINT commands but they could actually be two completely different commands and in fact you could have more than just two, the only limitation to how many you have is that you cannot exceed the length of three screen lines!

Cutting Up Strings

One of the very useful functions of Atari BASIC is the ability to print, examine or alter any part of a string Suppose that I have a string variable (call it A$) which holds the words MY ATARI COMPUTER and I want my program to print the middle word 'ATARI In order for it to pick out part of A$ I must tell it two things the positions of the first character and the last character The A of ATARI is the fourth character in A$, and the I is the eighth so I would write this as A$(4 8) Type NEW, then enter the following example to demonstrate how this is used in practice

```
10 DIM A$(20)
20 A$ ="MY ATARI COMPUTER"
30 PRINT A$
40 PRINT A$(4,8)
```

The result should look like this –

```
MY ATARI COMPUTER
ATARI
```

Now without erasing the program so far enter the following additional lines

```
50 A$(4,8)="HOME "
60 PRINT A$
```

Type RUN and you will see the following on your screen –

```
MY ATARI COMPUTER
ATARI
MY HOME COMPUTER
```

So you can see that you can also force a different string into part of another string as well, a very useful feature indeed – as you will no doubt discover in time! Something which you should note here is that the spaces in the strings actualy count as a character I said earlier that you can use a number variable in place of a literal, or fixed number and this is certainly true here Type NEW and type in and then RUN the following program then you will see what I mean!

```
10 DIM A$(20)
20 A$="MY ATARI COMPUTER"
30 A=1
40 PRINT A$(1,A)
50 A=A+1
60 GOTO 40
```

The program should print to your screen the following –

```
M
MY
MY
MY A
MY AT
MY ATA
MY ATAR
MY ATARI
MY ATARI
MY ATARI C
MY ATARI CO
MY ATARI COM
MY ATARI COMP
MY ATARI COMPU
MY ATARI COMPUT
MY ATARI COMPUTE
MY ATARI COMPUTER
ERROR 5 AT LINE 40
```

Notice the error message If you try and make the computer do something that is not possible it tells you so and in a very precise way The computer tells you what line number the problem occured on and in this instance the error occured on line 40 You are also told what went wrong, but there are around 50 different kinds of errors so rather than explain it to you the computer prints an error number – in this example error 5 Now you will need to look through your BASIC REFERENCE MANUAL and find the table of ERROR CODES You will then see that ERROR 5 means STRING LENGTH ERROR

This type of error usually occures when you try and reference part of a string which is beyond the current length of this string and so doas not exist If you look at line 40 you will see that the offending number must be in the variable A So let us find out what value caused the error Type –

```
PRINT A
```

You will find that it holds the number 18 Since in line 20 you have only placed 17 characters into A$ it is illogical to try and print up to the 18th – hence the program crashes with error 5

Hopefully by now you are reasonably familiar with strings and numbers, and are able to write short programs to manipulate and examine them but suppose that you would like your program to ask you for a number or a string – how is this done?

Using the INPUT Command

The INPUT command is one of the most valuable commands available to you It forces the computer to pause and wait for you to type a number or string on your keyboard and it will continue with the program as soon as you press the RETURN key Try the following program –

```
10 INPUT A
20 PRINT "THE NUMBER TYPED WAS " A
30 GOTO 10
```

You could consider the INPUT command as being the opposite to the PRINT command They both work with strings as well as numbers, as you will see by trying the following –

```
10 DIM A$(100)
20 INPUT A$
30 PRINT "YOUR STRING IS " A$
40 GOTO 20
```

Notice that when your program is waiting for you to type something the computer prints a question mark (?) as a prompt Whatever you type then goes straight into the variable you mention with the INPUT command. Something which you must remember is that if your program asks for a number to be entered (e g using INPUT

Continued on page 13

REVIEWS

MR. DO!

48K Cassette £9 95 Disk £14 95
from US Gold
Reviewed by Gary Cheung

Starring as a cute little clown, I went out on my daily round of cherry picking As usual, those little monsters are out to stop me, but I am determined and armed! Clutching my trusty power ball, I push my way through the earth and set up my trap I can almost smell those meanies coming up the tunnel as I push a big red apple upon them Whilst savouring that satisfying little victory, another monster creeped into view and, with what can only be pure reflex, I hurled my power ball and sent it to another dimension!

Having flattened some monsters, I came upon the lair and discovered one of the fabled centre treats Wow! a dish of ice-cream yum yum! Soon after I had polished this off I heard some strange marching sounds Oh Oh! Its the Boss Monster and It's dreaded henchmen I've got to run!

Mr Do! is a very good translation of the arcade game of the same name As you may have gathered, its a digging game You pick up cherries as you tunnel through a cherry field, while being pursued by various monsters. Points are gained by picking cherries, eating the centre treat, killing the monsters and picking up the legendary Lucky Diamond Extra Mr Do!"s are awarded, not by points but when all five types of Boss Monsters are killed Boss Monsters are marked with the letters E, X, T, R or A hence they are also known as EXTRA letter monsters!

Regular Monsters move along pre dug tunnels and can be killed by your power ball or a falling apple Diggers are the only monsters that dig tunnels (just as wall!) and can be killed in the same way Beware Regular Monsters can turn into Diggers! A Boss Monster appears at every 5000 points and when you eat the centre treat, the henchmen follows! The Boss and its dreaded henchmen cannot always be destroyed by falling apples, but killing the Boss will change its henchman into apples

Mr Do! features very good graphics and nice little tunes. There are many nice touches such as your Mr Do! changing poses as he moves the Regular Monsters eyes and legs move and the henchmen wobble as they chomp on the applas When an extra Mr Do! is awarded, a cartoon shows your Mr Do! strolling up to a monster and hurling a power ball at it, the monster starts to sweat (it sure looks like sweat!) and waves a white flag in surrender! Datasoft claims that there are virtually a limitless number of screens but I found that screen 11 is nearly identical to screen 1 screen 12 to screen 2 and so on

Mr Do!

There is only one gripe, that is the cassette version loads in five parts and takes over 16 minutes! Another thing (did I only say one? Sorry!) the disk version cost £5 extra!! Having said all that, it is a very addictive game and beats Pacman and Dig Dug hands down I can recommend it to any arcade gamer who likes a little strategy

A TIP FOR BEGINNERS.

To set up a trap, dig straight upwards to a position adjacent and one step below an apple Now pass under the apple and it will fall behind you. Then turn around and push the apple slightly so that it just sticks out into the vertical tunnel To get the maximum points wait until lots of monsters are in the tunnel then push the apple onto them

Top-DOS. The Ultimate DOS

32K Disk, Price £37 95
Reviewed by Matthew Tydeman.

I've always been a great fan of Atari's menu driven DOS 2 0s It is simple and very user friendly Atari's DOS 3 0 was a step back for Atari in my eyes, as it was for many other people I know, who have encountered it.

I use DOS 2 0s regularly and I like all my DOS systems to be compatible with each other (thats why DOS 3 0 was out of the window straight away!) I occasionally use OSS DOS XL, or ICD's Sparta DOS (with Ultra Speed I/O Processes) Both of which are compatible with DOS 2 0s

DOS 2 0s is good for its age (and is a great improvement on DOS 1 0) For many a year now, there has been a long wait for a DOS 2 0s update, something which has multiple compatibility and many new features to help the programmer get the most from his disk based system

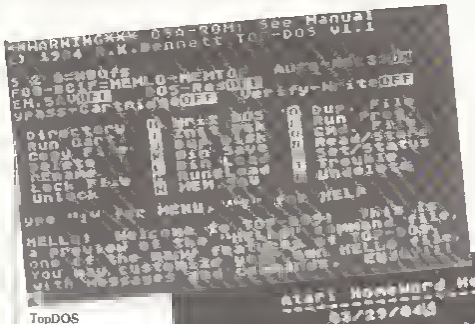
Top-DOS, from Eclipse Software, has done just this Top-DOS is a manu driven DOS with many more features than Atari DOS 2 0s. I counted 43 more commands. Many are features to help the lazy user, but some are a real help to programmers who need to get the best from their DOS I shall briefly outline some of the major commands which attracted me to Top-DOS

The directories are alphabetically listed and numbered for ease of use. You can search a directory for all the files 'beginning with, for example, A and specify to copy them to another disk, print them to printer or print them to screen A nice feature which alleviatas the necessity of exiting to Basic or Assemblar, to load a file to see its coding.

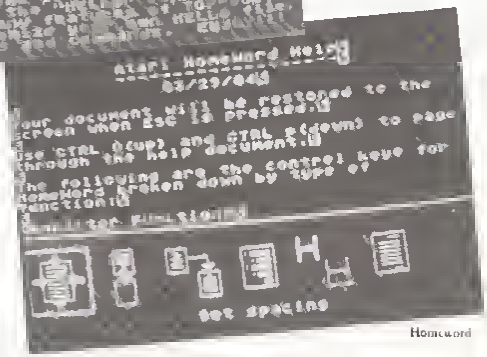
You can specify the length of the file name, and the number of columns in the menu, (1 to 6) A trouble command lets you see an error message in English No more codes to learn! You can boot the RS232C Handler from DOS, choose Resident DOS, or Non-resident DOS, change bytes in memory using the Editor and Monitor, already installed in memory when Top-DOS is booted, and there is one of my favourite commands, Undelete file

Top DOS is divided into two parts the first part being the main menu, which is similar to a DOS 2 0s menu but with 12 more commands. The second part is a customiser menu with around 22 commands. This customiser manu is a very useful part of the DOS It allows you to make your own version of DOS, with your own pre-coded requirements Something similar to that of a Printer coding disk, with all the Printer codes set up for your particular Printer

This customiser section does'nt write codes to the disk however it makes alterations to the DOS in memory and then



TopDOS



Homeword

writes the new modified version out when you are finished Some of the options available are Change Resident DOS Verify write Single/Double/Quad Density File Buffers Drive Control bytes Bypass Cartridge, Disk Buffers System Drive Number Margins Locations in memory and even change the prompt cursor

One feature I have not experimented with yet, is the Modify Drive Control bytes This useful feature can only really be utilised on Percom Rana Concord, Trak and Indus drives These drives have transferable bytes Twelve bytes can be modified to support Double Density, Sector Count and other nice features such as Access Times

Something I enjoyed using was the revised copy option You can merge copies of files together and save out as a complete file on another disk Also, I enjoyed the new format feature Not only can you specify how you want a disk formatted, but you can format a disk in 3 seconds! It does this by cheating a little in that it only formats the VTOC The formatting range can also be altered (720 is the default) to 934! Top-DOS has its own help feature included (far easier to use than DOS 3 0s) and now contains sub-files There are only two files that make up Top DOS DOS SYS and DUP SYS Top DOS even has its own AUTORUN.SYS creator called, appropriately, 'HELLO' Something I am yet to experiment with fully

Above all, Top-DOS is what I would call the Ultimate DOS When I first looked at it I thought it was complicated and hard to use, but I was very wrong! There is nothing more simple than operating Top-DOS I am sure that my personal enthusiasm plays a biased role in this review, but I hope it still gives you some idea of the capabilities of the 'TOP' DOS

Homeword

48K Disk, Price £54 95
Reviewed by Matthew Tydeman

Homeword is one of the latest word processors on the market and is somewhat different in appearance to that of many others that are around at the moment. Homeword is supported by a full range of Icons (a small illustrated picture of the performing function) In a sense Home word is not menu driven although text prompt lines accompany each Icon on display Within the main Icon menu are illustrations of a disk a printer a filing cabinet and writing paper As you select each Icon you enter another Icon menu linked with the title chosen, i e if you select the filing cabinet menu you will enter a sub-Icon menu which contains a number of filing cabinets each with a different function, such as Merge File Copy File Erase File, Get File Save File, etc.

Because of the Icons, the screen is divided into two display regions The upper half is for editing (only 16 lines) and the lower half is the region in which the Icons are shown and also the memory situation and disk storage situation are graphically represented with bars In the bottom right hand corner is an 80 column display, each letter being shown on this mini page as a Graphics 8 pixel, even a flashing cursor is included

Other than these extraordinary functions Homeword operates like any other word processor, but with a few limitations Block Delete/Copy File Search/Replace Justify Merge Include and Move are all there but are hard to access and utilise Each sub-Icon menu is called from disk in order to save memory space (essential in a word processor), and this calling of each menu from disk takes some time, which tends to prolong completion of the letter/document. Selection of the desired Icon is complicated at times and one false move with the cursor will load the wrong Icon menu

A feature I did not like much was that the Icons were too extensive once you entered one Icon menu, you had to go on to another and another and then yet another This aspect, I feel makes user friendly operations into frustrating time consuming annoyances and this is what Icons should nt be! The main reason for having Icons is to simplify and make things easier, In Homeword s case, for example, sometimes you have to go three or four layers to get back to the main Icon menu Very upsetting

A good feature however is that you can customise your own printer codes to match your printer Included with the program is the Customiser File, which is quite easy to use. All the editing features are the same as normal screen, as is the ever popular CONTROL and 1 letter formatting, i e CONTROL + I sets the page indent. But note that these codes are different from Atariwriter & Letter Perfect (whose codes are similar)

A nice touch is the Preview Document This is presented differently from the usual preview division in word processors in the sense that it is in full 80 column format. A nice feature to use but unfortunately it is only in the preview option and not the edit option still at least you won't have to worry about mistakes as you can see your whole document scroll up the screen just the way it will be printed.

The program would be good for youngsters they can t really go wrong with the Icons For those who don t like the idea of Icons there is a system to cut out all Icon operation This system just stops cursor control for the selection of Icons it does not however cut out the different loading processes it just enables you to key code instead, i e press Control + F for Filing Cabinet

Of course when using this key-code method for controlling the Icons, there is inevitably a lot of key-codes to remember This can be eased by the use of quick reference cards which would have all the codes marked on All the codes are also stored on the master system diskette as a printer file so you cannot really go wrong in this respect

The overall program instructions are quite extensive, but not very helpful when it comes to problems which often happens when playing around with the various Icon menus The program is average when it comes to judgement. It is not really what it is made out to be! Many people have got the impression that the program is along the lines of a full pull-down menu system, this is not the case! But its a pretty good attempt.

An average program for an above average price something Sierra should think about, I feel. I could not get to grips with this claimed 'Easy to use' word processor. The idea of Icons on an eight bit machine has not really appealed to me The use of Icons on a sixteen bit micro shall I'm sure!

GO FORTH

by Paul Blackmore

This article is an introduction to one of the more popular alternatives to BASIC as a programming language

FORTH as it is generally known today is based on FORTH 79 the most usual implementation being to the Fig FORTH model This is a minimum specification most commercial implement ations of FORTH have enhancements to take advantage of a particular computers capabilities There are a number of different FORTHs for the ATARI mostly disk based, although English Software have recently produced a cassette version There are currently three disk versions available FigFORTH by ATARI (APX) approx £7, Q S FORTH by Quality Software and ValFORTH by Valpar International As FORTH was designed to operate with disk based systems this article is in general talking about disk versions of the language.

FORTH Advantages?

Firstly, FORTH is significantly faster than BASIC in most applications This is because FORTH source code is compiled and not interpreted at run time (this is not strictly true as FORTH has a small run time interpreter) Secondly FORTH code is compact, allowing larger programs in the available memory A FORTH word, once defined takes only two bytes of memory when subsequently used inside another word definition.

What is FORTH?

The most fundemental structure in FORTH is the stack. Anyone who has delved into the intricacies of assembly language will understand the concept of the stack For the uninitiated, the stack can be likened to a pile of papers each sheet of paper in the pile having an item of data written on it. Obviously the last sheet put on the pile is the first one to be taken off the top commonly known as LIFO (last in first out) and to get at something halfway down the pile you have to move the others off the top first.

One of the most apparent differences between FORTH and other languages is the use of Reverse Polish Notation (RPN) This is usually one of the earliest stumbling blocks when learning FORTH To my knowledge no other language uses RPN, although some of you may know it from some of the early scientific calculators. Basically it works in this manner

On a normal calculator the calculation –

6(5+(4*3))=

Would entail a series of keystrokes something like this.–

4 * 3 = + 5 = * 6 =

In FORTH, this calculation becomes – 6 5 4 3 * + *

The ' in FORTH the print stack command, is the equivalent of hitting the equals key on the calculator

The top two numbers on the stack (3 & 4) are multiplied by the first '*' leaving the result on the top of the stack. The '+ then takes this result and adds it with the next number down the stack the 5 and leaves the result of that calculation on the stack Finally the '*' takes these two remaining numbers and multiplies them together The last operator the '.' prints the number left on the top of the stack.

It can be seen from this example that the stack can be loaded with data and then each item of data can be accessed sequentially by the mathematical operators. There are many commands within FORTH that allow manipulation of data on the stack so that extremely complicated calculations can be performed without vast numbers of brackets in fact without the use of brackets at all!

This brings me to FORTH's most powerful feature the ability to define your own command words The words that you define can then in turn define other words which ultimately becomes your program There is almost no limitation on what words do or even on what they are called, (you cannot have a word with a space in its name or certain control characters) In fact, some of the standard words in FORTH are nothing more than symbols, e g. @!–> etc

For instance you could define the word AVERAGE to calculate the average of three numbers. It could look something like this –

```
AVERAGE + + 3 /
```

The ':' and the ';' mark the start and finish of a word definition This word needs three numbers on the stack, it adds them together divides by three and prints the result. It would be used in the form –

99 338 225 AVERAGE

This would print the result '220 Ok (the 'Ok.' is like BASIC printing READY)

Note that the result is 'integer only', i e there are no decimal places in the answer With a different definition of AVERAGE it is possible to get an answer with a remainder –

```
AVERAGE + + 3 /MOD     "/3"
```

The "" prints the text between the quotes. This would print the result. –

99 338 225 AVERAGE 220 2 /3 Ok.

Standard FigFORTH does not support floating point maths If you have ATARI's FigFORTH or ValFORTH however this has been added as an extension to the system It should be noted that the use of floating point maths in FORTH is not much quicker than it is in BASIC, it uses the same OS routines!

The FORTH command words available allow you to have the very fashionable 'structured programming' Apart from words that allow manipulation of memory there are loops such as DO–LOOP DO–+LOOP conditional structures like IF–ENDIF IF–ELSE–ENDIF, BEGIN–AGAIN BEGIN–UNTIL BEGIN–WHILE–REPEAT and the extremely powerful <BUILDS–DOES> construct. With <BUILDS–DOES> words can be created that define defining words (read that again it does make sense!)

In ATARI FORTHs there are command words on the system disk for graphics, sound, printer handling an editor and a complete assembler vocabulary allowing the use of assembly language in a FORTH program ATARI's FigFORTH and ValFORTH also support RS232 interfaces DOS I/O and floating point maths

Writing a FORTH Program

Writing a FORTH program can be done in two ways You could, as in the case of the short examples given here, just type them into the FORTH dictionary directly from the screen editor Alternatively the better way, aspacially when the word definitions are large is to use the FORTH editor to write 'screens' for later compilation. In fact this is the only way that you can save your word definitions for later modification unless you save the complete FORTH system containing the new words.

The standard FigFORTH editor is

perhaps FORTH's most notorious feature, it is not as easy to use as the standard full screen editor used by BASIC. Some versions of FORTH have tried to remedy this notably Q S FORTH, and provided an editor more like the one we are all used to. Unfortunetely most people have to use the RAGSDALE editor provided, (which incidentally is written in FORTH) unless you wish to write your own editor of course!

Standard ATARI FigFORTH works by treating the disk as blocks of 'virtual memory'. This means that FORTH sees a single density disk as 90 external blocks of 1K memory. Other versions of FORTH use smaller size blocks usually 1/2K per screen. This allows more screens per disk and also saves on wasted disk space. Once you have written your program onto these screens using the editor you can 'LOAD them onto the FORTH dictionary using – 10 LOAD(remember, everthing is backwards!)

The 'program' written on screen 10 will be taken off the disk and compiled into the dictionary. If screen 10 ends with the load next screen command, –>, then screen 11 will also be loadad and compiled. When all this has finished, your command words will be in the dictionary and can be listed with list vocabulary command, VLIST. Your program can now be run, just type its name!

Dictionaries & Vocabularies

You may have noticed that I have been talking about dictionanes and vocabulanes and have not explained what they are. Well a dictionary is a list of words (surprise surprise!) found in a vocabulary. To explain the words found in a French dictionary are the words in the French 'vocabulary' i.e. the French language. The same thing applies in FORTH. The words in a FORTH dictionary are the words that FORTH can understand, if you change the vocabulary to something else e.g. ASSEMBLER then the vocabularies are a bit smarter than this everything defined in a vocabulary will eventually trace back to the FORTH vocabulary the words are effectively translated for you by the system. What this means is that if an application does not have a word defined in the CURRENT vocabulary then FORTH will trace back and try to find it in the FORTH vocabulary.

To illustrate the use of vocabulanes, suppose that FORTH is the 'CURRENT' vocabulary i.e. the one that we are compiling our word definitions into, we could define a word to tell us which vocabulary we are working in, thus –
WHERE " defining into FORTH".

Now we create a new vocabulary and set it to be the current vocabulary by typing – PROGRAM DEFINITIONS

From now on all new word definitions will go into the PROGRAM vocabulary. If we redefine WHERE thus –
WHERE " Defining into PROGRAM".

A message "WHERE is not unique" is printed because the system knows about the definition of WHERE that we just put into the FORTH vocabulary. If we now type – WHERE

We get the message 'Defining into PROGRAM'. Now type
FORTH DEFINITIONS
WHERE

We get the message 'Defining into FORTH'. The uses for vocabularies are endless, you could write adventures, telephone and mailing lists, language translators, database systems, word processors or even disk operating systems all based around the use of different vocabulanes.

Of necessity, this article has not been too technical, nor attempted to teach FORTH programming, but I hope that it has at least given you an insight to the possibilities of using an alternative language like FORTH.

A final word of caution to all possible converts: because of its total flexibility FORTH can be quite hostile to programmers with untidy minds. The lesson is: forget all those nasty little habits you used to get away with in BASIC and learn the correct wey of doing things!

Starting from Basics
Continued from page 9

A), if you type a non – numeric string such as a name (or even in fact by pressing the RETURN key with no number) BASIC will give an error (ERROR 8 – INPUT STATEMENT ERROR) and your program will stop.

```
10 DIM C$(20) S$(20)
20 PRINT "CHRISTIAN NAME": INPUT C$
30 PRINT "SURNAME": INPUT S$
40 PRINT "AGE", INPUT A
50 PRINT
60 PRINT "YOU ARE ",C$;" ";S$
70 PRINT "YOUR AGE IS ";A,"YEARS"
80 PRINT "(ROUGHLY ";A*365;" DAYS!)"
90 PRINT: PRINT
100 GOTO 20
```

Type NEW and enter the above program then type RUN. Answer the questions. Notice how I have used a semi colon after the PRINT commands that give the prompt for the following INPUT command (e.g. on line 20). This is so that the cursor remains on the line immediatly after the prompt. If you wanted the INPUT command to continue on the next line you would omit the semi-colon. Remove the semi-colon on line 20 and see how this changes the way the program works.

Suppose that you decide that you went to insert another program line between lines 60 and 70. Now you should see the advantage in using line number references in increments of 10: for you can simply enter your new line as number 65, the computer will then insert it between lines 60 and 70 in its memory. Try typing the following line –

```
65 PRINT "INITIALS ARE ",C$(1,1);S$(1,1)
```

Now RUN the program again and you will see that the extra line does indeed print your initials on the screen. Type LIST to see that the new line is placed in the correct position.

Using LIST

The command LIST can be used in several ways. You have seen that LIST when used on its own will cause the computer to list your entire program to the screen. This is great for small programs but suppose that you are altering a line in the middle of a large program, you would have to wait patiently while the computer lists all the wey through until it reached the line you want. There is, however, a much more convenient way of looking at a single line, you simply type LIST 90 to list line 90. You can even list sections of a program if you wish, by simply typing (for example) LIST 50,80. This would list every line between (and including) 50 and 80.

LIST can also be used to any other device you have. For example, to list to a printer you would type LIST "P:" or LIST "P:", 50,80 for lines 50 to 80 only.

And Finally

If you are running a program which prints lots of data to the screen, or you are listing a long program to the screen, it is possible to 'freeze' the screen to give you time to read it. You do this by holding down the CONTROL key (on the far left of your keyboard) whilst you press the '1' key. The computer will instantly stop in its tracks and this gives you time to read the display. You tell it to continue by repeating the process. This only works whilst data is being printed to the screen – not with any other devices such as printers.

A useful feature of Atari BASIC is that virtually all of its commands can be abbreviated. For example, the command GRAPHICS 8 can be replaced with GR.8 and GOTO 10 can be replaced with G. 10. If you take a look at the manual supplied with your computer, you will see the abbreviated version next to each command. Try experimenting with them. Notice that, although you may enter your program line using the abbreviations when you actually LIST the program BASIC prints the full version! There is however, one exception to this. The command PRINT has a special abbreviation which stays in its short form when you LIST the program. It is the question mark (?). Try some of the programs again but this time use the question mark in place of PRINT.

That's all for now, but I will be back in the next issue to continue with our lessons. See you all again soon!

A RAMDISK FOR THE 130XE

by Ron Levy

Atari (US) have released a version of DOS (called DOS 2 5) which contains a RAMDISK, taking advantage of the 130XE's extra 64K of RAM Soon it will be freely available to users, but until that time, presented here is a short program which sits in page 6 of memory and which intercepts calls to drive 2 and instead of getting or putting a sector to drive 2 it transfers the data to the extra 64K of memory (the DOS 2 5 Ramdisk will access drive 8 -- Ed) The result is an incredible increase in speed when loading and saving programs or when working with files on the Ramdisk. Just like normal memory however the extra memory is lost when you switch off the computer so you must remember to copy all your files back to a real disk drive when you are finished. However since the Ramdisk only has 64K you will only be able to use up to sector 512 rather than the usual 709 (DOS 2 5 only allows access to 499 sectors)

I use K DOS rather than DOS 2, so this Ramdisk was originally intended to be used with K-DOS, but I have also included a patch to allow it to work with DOS 2 Using DOS 2 it is possible to configure the Ramdisk as drive 1 and then write DOS files to it. Then, if you are using BASIC and want to go to DOS you type DOS as usual, but now the DOS menu will be loaded well within 1 second, a startling speed!

CREATING THE RAMDISK

The first task is to type in and RUN the BASIC program Listing 1 This creates a file on your disk, named 'RAMDISK OBJ' This file is a BINARY LOAD file and it contains the main Ramdisk program The same program will work for both K-DOS and DOS 2 and it will load onto the bottom half of page 6 so it does not conflict with the operating system It won't however work on its own since DOS will not use it we now have to alter our DOS to make it call our routine before it transfers sectors of data to and from disk

For disk sector transfers both versions of DOS use a routine in the first part of DOS This routine calls the serial bus routine in the operating system with a JSR $E459 instruction but we must change this to point to our routine with a JSR $600 This simply involves changing the two appropriate bytes and in DOS 2 these are locations $7A3 and $7A4 (decimal 1955 and 1956) The locations in K-DOS are $7AD and $7AE (decimal 1965 and 1966) The contents of these two bytes must be changed to 0 and 6 respectively Although these locations can be altered using BASIC, by far the quickest way is to have a further binary load file and consequently I have included a program to create such a file for you. Program 2 is for use with DOS 2 and program 3 for K DOS Each will create a binary load file named 'PATCH OBJ'

```
KF 1 REM .........Program 1.........
IN 2 REM This progrem creates a binary
BT 3 REM load file called RRROISK.OBJ
SV 4 REM This will be the main ramdisk
BF 5 REM progrem and it lives in page 6
RL 6 REM
JG 10 OPEN #1,8,0,"D:RAMDISK.OBJ"
KM 20 TRAP 200
IL 100 READ X:PUT #1,X:GOTO 100
KY 200 CLOSE #1
HC 210 PRINT "Complete."
TJ 300 STOP
FO 1001 DATA 255,255,0,6,140,6,173,1,3,20
    1,2,240,4,32,89,228,96,173,10,3,133,21
    2,173,11,3,133,213,41,254,208,77,173
KJ 1002 DATA 4,3,133,214,173,5,3,133,215,
    32,100,6,32,125,6,160,127,173,2,3,201,
    82,208,22,166,213,142,1,211,177,216
ML 1003 DATA 162,241,142,1,211,145,214,13
    6,16,239,160,1,76,102,6,201,80,240,4,2
    01,87,208,22,177,214,166,210,142,1
WI 1004 DATA 211,145,216,162,241,142,1,21
    1,136,16,239,169,1,76,102,6,160,1,140,
    3,3,96,165,213,106,165,212,41,128,106
NG 1005 DATA 74,74,74,74,9,225,133,210,96
    ,165,212,24,41,127,106,9,64,133,217,16
    9,0,106,133,216,96
```

USING WITH DOS 2

Boot up your 130XE with the DOS 2 MASTER DISKETTE, (you must leave BASIC installed, and type DOS to go to menu) Insert a fresh diskette and use option I to format it, then use H to write DOS files to it. Now return to BASIC (option B)

Type in and SAVE program 1, then RUN it. This will create the file RAMDISK.OBJ' Now type in and RUN program 2, which will create PATCH OBJ', the file that will overlay and 'steal' the serial interface vector in DOS You must

```
KG 1 REM .........Program 2.........
IN 2 REM This program creates a binary
MH 3 REM load file called PRTCH.OBJ
KK 4 REM to force DOS 2 to call the
OM 5 REM ramdisk routine.
ML 6 REM
WJ 10 OPEN #1,8,0,"D:PRTCH.OBJ"
KM 20 TRAP 200
IL 100 READ X:PUT #1,X:GOTO 100
KY 200 CLOSE #1
HC 210 PRINT "Complete."
TJ 300 STOP
DO 1001 DATA 255,255,163,7,164,7,0,6
```

now return to the DOS menu using the command DOS When the menu appears type L for BINARY LOAD The computer will ask for a file name and you should enter RAMDISK.OBJ Now repeat the process for the file PATCH OBJ You now have a Ramdisk in your system, set up as drive 2 As with a normal disk, the first thing you must do is to format disk 2 using option I At this stage you can even look at the directory of the Ramdisk, which will of course show 709 free sectors although you are only able to use up to 512 sectors. How you use your Ramdisk depends upon the tasks you need to perform, but you could now return to BASIC and LOAD programs from drive 1 and SAVE them to drive 2 If you are developing your own programs you could use the Ramdisk to store any utility routines or files that you use regularly and the easiest way of doing this is to use the file copy facility (option C) on the DOS menu The best way of doing this is to create a normal disk full of your utility routines and programs, and copy them to the Ramdisk using the copy facility with wildcards for the filenames, e g. D1 * *,D2 * *

Another way of using the Ramdisk is to configure it as drive 1 then format and write DOS files to it. This way you can flip between BASIC and DOS at incredible speed In fact, even when you use a MEM SAV file DOS will appear within 1 second!

```
LE 1 REM .........Program 3.........
IN 2 REM This program creates a binary
NH 3 REM load file called PRTCH.OBJ
CV 4 REM to force K-DOS to call the
DN 5 REM ramdisk routine.
ML 6 REM
WJ 10 OPEN #1,8,0,"D:PRTCH.OBJ"
KM 20 TRAP 200
IL 100 READ X:PUT #1,X:GOTO 100
KY 200 CLOSE #1
HC 210 PRINT "Complete."
TJ 300 STOP
QK 1001 DATA 255,255,173,7,174,7,0,6
```

Unfortunately though, as the program stands you cannot write DOS files to drive 2 and then convert the Ramdisk to respond as drive 1 since DOS does a status check on the drive via a different route first. Consequently I have included program 4, which will provide you with yet another binary load file This file when loaded with option L, will perform the equivalent of a POKE 1540 X where X represents the number we want our Ramdisk to respond to Once you have changed the Ramdisk to drive 1 you can write DOS files to it (option H) and then return to BASIC You would then probably want to switch your normal disk drive to respond as drive 2

USING RAMDISK WITH K-DOS

The method here is essentially the same, except for a few points which I shall explain. Run program 1 as normal, but instead of program 2, use program 3. There is no need to write DOS files to the Ramdisk since with K-DOS, the utilities package is permanently in memory. K DOS performs a check on the destination location of binary load files to ensure that you do not accidentally corrupt DOS, so to force our patch file in, you must use the /PUT switch thus –

LOAD PATCH.OBJ/P

The DISKDUP.SYS program supplied with K-DOS, will not work with the Ramdisk unless a small modification is made.

First, insert a K-DOS master disk with DISKDUP.SYS on it and go to DOS. Type – LOAD DISKDUP.SYS/M

Then –

A 42DE A9
A 42DF 01
A 42E0 EA
SAVE DISKDUP.RAM 4000 47C0 4000

You will now have created the disk file named DISKDUP.RAM, which you call with the command,

RUN DISKDUP.RAM

Or, if you copy it to your Ramdisk –

RUN D2:DISKDUP.RAM

Although K DOS features abbreviated commands, I have shown them in full to ensure clarity.

```
10 ; ******************************
20 ;*       RAM-DISK             *
30 ;*  A Virtual Disk Storage    *
40 ;*  System Which Uses The     *
50 ;*  Extended RAM Facility     *
60 ;*    On The ATARI 130XE      *
70 ;*       BY RON LEVY          *
80 ; ******************************
0100 ;
0110 ;   Disk Interface Equates.
0120 ;
0125 SIOV    =$E459
0126 DDEVIC  =$300
0130 DUNIT   =$301
0140 DCOMAD  =$302
0150 DSTATS  =$303
0160 DBUFLO  =$304
0170 DBUFAI  =$305
0180 DAUX1   =$30A
0190 DAUX2   =$30B
0200 SECTSEC =$52
0210 PUTSEC  =$58
0220 FORMAT  =$21
0300 RAMCTL  =$D301
0310 T1      =$D4
0320 T2      =$D5
0330 T3      =$D6
0340 T4      =$D7
0350 BPL     =$D8  RAM block position
0360 BPH     =$D9 for a given sector.
0370 RAMSEL  =$D8   Bank select mask.
0400 NORMAL  =$41  "Normal" RAM MASK.
1000         *=$600
1010 BEGIN
1020         LDA DUNIT  Check the
1030         CMP #$02   drive number.
1040         BEQ @1
1050         JSR SIOV  Go to SIOV if
1060         RTS    ite not for RAA.
1070 ;
2000 ;
2010 ; Main routines here. ;
2020 ;
2160 ;
2170 @1
```

```
2180         LDA DAUX1  Transfer the
2190         STA T1     sector number.
2200         LDA DAUX2
2210         STA T2
2212         AND #254   Check for
2214 ;                  sector number
2216 ;                     too high.
2218         BNE EXIT
2220 @2
2230         LDA DBUFLO Transfer the
2240         STA T3          buffer
2250         LDA DBUFRI      pointer.
2260         STA T4
2270 ;
2280         JSR RAMBANK
2290         JSR RAMPOS
2295         LDY #127
2300 ;
2310         LDA DCOMND
2320         CMP #GETSEC
2330         BNE PUT
340 ;
2350 GET
2360 ;                  The GET sector
2362 @1      LDX RAMSEL     command
2364         STX RAMCTL     routine.
2370         LDA (BPL),Y
2372         LDX #NORMAL
2374         STX RAMCTL
2380         STA (T3),Y
2390         DEY
2400         BPL @1
2405         LDY #$01
2410         JMP EXIT    Done !!
2420 ;
2430 PUT
2440         CMP #PUTSEC
2450         BEQ P1
2452         CMP #$57
2454         BNE EXIT
2460 ;
2500 ;                  The PUT sector
2510 P1      LDA (T3),Y     command
2512         LDX RAMSEL     routine.
2514         STX RAMCTL
2520         STA (BPL),Y
```

```
2522         LDX #NORMAL
2524         STX RAMCTL
2530         DEY
2540         BPL P1
2542         LDA $$01
2550         JMP EXIT    Done !!
2590 ;
2599 ;
5000 EXIT
5010         LDY $$01
5020         STY DSTATS
5090         RTS    Done and Bye !!
9000 ;
9010 ; Select A RAM Bank. ;
9020 ; T1 = low  ;  T2 = high
9030 RAMBANK
9040         LDA T2  The high byte.
9050         ROR A
9060         LDA T1  The low byte.
9070         AND #128
9080         ROR A
9100         LSR A
9110         LSR A
9120         LSR A
9130         LSR A
9140         ORA #225 Set other bits.
9150         STA RAMSEL
9190         RTS          Done !!
9198 ;
9199 ;
9200 ;
9210 ; Calculate the data block
9220 ; position in RAM.
9230 RAMPOS
9240         LDA T1  Sector low byte.
9245         CLC
9250         AND #127
9260         ROR A
9265         ORA #64
9270         STA BPH
9280         LDA $$00
9290         ROR A
9300         STA BPL
9310         RTS          Done !!
9390 ;
9399 ;
```

KEYO KEYO KEYO KEYO KEYO

The Ultimate Checksum Program

by Eddie Taw

By far the most frequent comment made to us by readers of Monitor is 'Why don't you have some kind of program entry checker to help us type in your listings?' Well, never let it be said that we would ignore such a regular request from our readers, so here is KEYO!

You may have noticed that in this issue each listing has a two character code before each line number. This is the checksum code which KEYO will use to determine whether or not you have typed the program line correctly. You might also notice that it is similar to the checksum letters used in several other magazines. In fact, you can actually use our program to enter the listings in those magazines or perhaps use their checksum program to enter the programs in MONITOR if you prefer!

What makes KEYO different?

KEYO works by saving your program onto cassette or disk as you enter it at your keyboard. It also asks you for the checksum code for each line and compares this to the line you have just entered. We feel that KEYO offers several advantages over other checksum entry utilities —

a) Since KEYO requires you to enter the given checksum code as well as the line and rejects the line if it does not match, there is no danger of you getting a wrong line as there could be if you had to check every code yourself.

b) Since KEYO saves to cassette or disk as you type, if the computer crashes for some reason (e.g. the wife/sister/brother unplugs your computer to use the hoover!) then don't worry - you will only lose a very tiny part of the program you have entered (128 characters at the very most).

We hope that you find KEYO helpfull but PLEASE remember that we would like to know of any difficulties you encounter whilst using it, or indeed any improvements you think we could incorporate. Feedback from users is of vital importance if we are to improve the service to you - our readers.

```
EI 1 REM ************************************
NH 2 REM
TI 3 REM KEYO
NJ 4 REM
UG 5 REM BY EDDIE TAW
ML 6 REM
YN 7 REM FOR MONITOR MAGAZINE
NN 8 REM
EO 9 REM ************************************
NT 10 DIM C$(2),CHK$(3),F$(15),L$(132)
BX 20 ? CHR$(125);" KEYO Handityper --
   By Eddie Taw. "
KI 30 ? "          For MONITOR Magazine."
VF 40 PRINT :PRINT
QB 50 ? "Do you want Instructions Y/N ";:
   INPUT L$:IF L$="Y" THEN GOSUB 10000
YJ 100 ? "Output file ";:INPUT F$:IF F$(1
   ,1)="I" THEN 50
XC 120 IF F$(1,1)="C" THEN ? "Insert TAPE
   and press RECORD & PLAY"
XB 200 OPEN #1,8,0,F$
EG 210 IF F$(1,1)="C" THEN FOR I=1 TO 130
   :L$(I,I)=CHR$(32):NEXT I:PRINT #1;L$
VQ 220 SOUND 0,0,0,0
VN 300 ? "Type the CODE...";:INPUT CHK$
UN 310 IF CHK$="END" THEN CLOSE #1:GOTO 3
   2767
ZK 320 IF CHK$="I" THEN GOSUB 10000:GOTO
   300
QU 400 PRINT "Type the PROGRAM line,"
AZ 420 POKE 85,8:? CHK$:POKE 85,1
IT 430 INPUT L$:IF L$="" THEN 300
NQ 440 IF L$(1,1)="C" THEN 300
SS 450 IF L$="I" THEN GOSUB 10000:GOTO 40
   0
GP 460 GOSUB 5000
VP 470 IF CHK$=C$ THEN PRINT #1;L$:PRINT
   "Ok. Line accepted.":PRINT :SOUND 0,0,
   0,0:GOTO 300
QE 500 AEA Error Routine.
OY 510 ? "Error ; Re-type program line."
YN 520 ? CHR$(253);:POKE 85,0
ZZ 530 ? CHK$:X=PEEK(84):? L$:POKE 84,X
JK 540 POKE 85,1
QF 550 GOTO 430
HP 5000 REM ...Calculate The Checksum...
XH 5010 S=0
FF 5100 FOR I=1 TO LEN(L$):S=S+I*ASC(L$(I
   ,I)):NEXT I
CE 5110 C=S-676*INT(S/676):HC=INT(C/26)
KQ 5120 LC=C-(HC*26)+65:HC=HC+65
YH 5130 C$(1)=CHR$(HC):C$(2)=CHR$(LC)
NV 5900 RETURN
BC 10000 AEM
KI 10010 FOR J=1 TO 4
KN 10020 GOSUB 11000:POSITION 2,4
PV 10030 ? "Instructions."
NK 10040 ? "------------"
HP 10050 AEGTONE 10100+J*100
UP 10060 FOA I=1 TO 14
EF 10070 NEXO L$:PRINT L$
SP 10080 NEXT I
GL 10090 ? :POKE 85,4:? "Press RETURN to
   continue...";:INPUT L$:NEXT J
FT 10100 GOSUB 11000:AETURA
JI 10200 DATA ,  KEYO will save your prog
   raa ae,it ie typed in ee first it euet
    ask
SP 10210 DATA where you want it esved to.
    It will,give the prompt 'Output Filen
   aas ?'
HA 10220 DATA ,  Cassette owners will typ
   e C: and,euet then place a good qualit
   y tape
WH 10230 DATA into their recorder (prefer
   ably C60),and praas the AECORD & PLAY
   buttone.
SZ 10240 DATA ,  Disk drive owners need t
   o epecify,a filenaan (ie D:PROS.LST) a
   nd then,insert a good disk.
UB 10300 DATA ,  KEYO will first aek you
   for the,two character code which you w
   ill,find next to each progrse line.
CX 10310 DNTN ,  Next It will aek for the
    progrse,line. If you wieh to go back
   to the previous prompt just prene
   AETURN,on ite own.  KEYO aill check th
   e,line and save it if it ie corrrect.
AX 10330 DNTA If it ie incorrect you aill
    be eble,to edit it or type C to chang
   e the
WC 10340 DATA code (you can type over the
    line).,
DZ 10400 DATA ,  To leave the progrse at
   any time,you simply type END when KEYO
    aeks,for a checkeum code.
SH 10410 DATA ,  You can continue enterin
   g your,progrse later by again RUNning
   KEYO.
BN 10420 DATA Caeeatte ownere simply lea
   ve their,tape in the player ee that KE
   YO can
YK 10430 DNTN create another file immedia
   tely after,the first but diek drive un
   ners will
BO 10440 DATA have to give a different fi
   lenaee,(eg PROG2.LBT).,,
AB 10500 DATA ,  To load the finished pro
   grae:-,,  CASSETTE owners aust first r
   ewind
SH 10510 DATA the tape and type NEW to er
   ese the,KEYO progrse. Then type ENTER
   "C:"
IB 10520 DATA for ss eany times ss you EN
   Ded and re-ran KEYO.,,  DISK DRIVE own
   ers should follow
HY 10530 DATA a slailar enquence except t
   hat you,aust type ENTER "D:PROG1.LBT"
   and
XA 10540 DNTN repeat thie for any other f
   ilenaeas,you have used.,
BE 11000 NEM
AB 11010 POSITION 2,3
RV 11020 FOA I=1 TO 20
HX 11030 PRINT CHR$(157);
CB 11040 NEXT I:DETURN
AO 32767 END
```

USER GROUP SOFTWARE

Software Librarian - Roy Smith

Due to demand from membets there are now two ways to get programs from the library The original scheme of exchanging 3 for 1 will still apply but now with an added bonus So the library rules have been extended to enable those membets who cannot write their own programs to gain access, and those that can to have a possibility of some reward for their efforts The extended library rules are as follows

3 FOR 1 EXCHANGE

1 Every program you donate to the library entitles you to three programs in return
2 The program you donate must be your original and not copied
3 Your donated program must be submitted on a cassette or a disk, programs in the form of print-outs will not be processed
4 If your program requires any special instructions they should be added in the form of REM statements within the program (or you may present them as instructions when the program is actually run)
5 BONUS Every program donated per quarter (between issues of the newsletter) will be eligible to be judged 'STAR PROGRAM' for that quarter This carries a prize of £10 which will be paid to the author from the club funds The programs will be judged by the Editorial Team and their decision will be final The Editorial Team are not eligible for the prize
6 The 3 FOR 1 exchange is only open to club members

DONATION SCHEME

1 Every club member will be entitled to ask for up to 3 programs per quarter from the library by donating to the club funds
2 If a member does not take his/her entitlement for a particular quarter it cannot be carried forward to the next quarter
3 A member can have more than one quarter's entitlement at one time (up to a maximum of 12 programs (1 year)), but then will be unable to ask for more until his/her credit quarters have been used. Note that odd numbers of programs will be counted in quarters i e if a member asks for 5 programs the first 3 will be that quarter s entitlement the next 2 will be the second quarter's entitlement and he/she will have to wait until the third quarter before he/she is entitled to any more Also note that having programs in advance will only be allowed if that member s membership covers the advance quarters
4 The donation fee will be £1 per program and is not refundable Cheques and Postal Orders are to be made out to the 'U K Atari Computer Owners Club'
5 Members must send in a blank cassette or diskette for the chosen programs to be recorded on
6 The DONATION SCHEME is only open to club members

Finally I would like to point out that some people omit to include return postage when donating to the library so please do not forget to include 30p worth of stamps to cover this

THE LIBRARY SOFTWARE SERVICE IS FOR MEMBERS ONLY

LIBRARY SOFTWARE TITLES

Listed below are the software titles received by members for inclusion in the library since issue eight was published. As the library now contains over 300 programs it is getting a bit too large to keep on printing the entire list. Eventually it would probably take over the whole magazine and there would be no room left for the articles and program listings. For those of you who are new members and do not know what is available from the library then send for a photocopy of the complete list which is available from the librarian. There is a small charge for this service to cover photocopying costs. If you would like a list please send 50p and a SAE for return.

Games

ASTERIOD BATTLE
by Mark Hutchinson – Belfast.
Pinball game created using Pinball Constructor
Runs in 48K min. Disk only
Not XL compatible.

****** STAR PROGRAM ******

CHASE
by Grahame Fairall – Oxon
Collect the money but don t get caught. Excellent game with 40 levels of play.
Runs in 32K min Disk only

DROPOUT
by Alan Williamson – Dunfermline
Shoot the dropping objects before they land or else!
Runs in 16K Cassette or 32K Disk min

FURTRADER
by Grahame Fairall – Oxon
Trade your furs, make money or starve Strategy game
Runs in 16K Cassette or Disk min

HAUNTED HOUSE
by Mark Hutchinson – Belfast.
Pinball game created using Pinball Constructor
Runs in 48K min Disk only
Not XL compatible.

MOTORMAZE
by Mike Barnard – Guisborough
Drive your bike around the maze to find the exit.
Runs in 32K min. Disk only
ACTION! program (ACTION! cartridge required)

NUMBER GRID
by Chris Simon – Mold
Two player game get the highest score by your quick reactions.
Runs in 32K Cassette or Disk min.

SUPER FRUIT MACHINE 2
by Grahame Fairall – Oxon.
Excellent graphics with all the usual features of a fruit machine.
Runs in 16K Cassette or 32K Disk min

UPS & DOWNS
by Stephen Taylor – London
Two player board game Get your ship or car to the top first collecting bonuses on the way Good graphics
Runs in 32K Cassette or 48K Disk min

Home Entertainment

BLOCKBUSTERS KIT
by Steve Tullett – Falkirk
3 person game simulating the score board of the TV Blockbusters game.
Runs in 32K Cassette or Disk min.

MICRO PUZZLER
by Stuart Mowles – Ipswich.
This program takes any of 13 Micropainter pictures and jumbles them into a jigsaw puzzle
Runs in 48K min Disk only
Requires both sides of a disk

PONTOON FRUIT
by Grahame Fairall – Oxon.
Fruit machine game
Runs in 16K Cassette or Disk min.

WORDCLUE
by Alan & David Williamson – Dunfermline
Word guessing game for 1 or 2 players
Runs in 16K Cassette or 32K Disk min.

Utilities

ATTENUATORS
by Robert Macleod – Glasgow
This program works out T network & PI network attenuator values
Runs in 16K Cassette or 32K Disk min.

CASSETTE INLAY PRINTER
by Stephen Taylor – London
Print cassette inlays on your 1020 printer
Runs in 16K Cassette or Disk min.

COMPOSED WRITER
by Larry Farmer – U S A
Extensive wordprocessor with full on screen editing print options and many other good features.
Runs in 16K min Disk only
Requires whole side of disk.
From ACE of Eugene, Oregon U S A.

COPYDISK
by Robert Gibbons – Holmes Chapel
Sector by sector disk copy utility (not protected disks)
Runs in 48K min. Disk only

DISASSEMBLER
by Mike Barnard – Guisborough.
Useful utility for the more advanced programmer
Runs in 32K min Disk only
ACTION! program (ACTION! cartridge required).

MULTIBOOT BOOTBASE
by Sol Negrine – Hockley
This program reads the directory of multiboot disks and can create a file directory and or print labels, compatible with APX Progfile
Runs in 48K min. Disk only

SCREENBASED WORDPROCESSOR
by Jeff Davies – Llandello
Simple wordprocessor utility includes Edit, Store & Recall and Print options
Runs in 16K Cassette or Disk min

SEARCH & SEARCH 1
by J Bennett – Newcastle.
Two programs which both look for an ATASCII string on a disk. SEARCH lists the sectors where the string is found. SEARCH 1 displays the section of the sector upto the occurrence of the string.
Runs in 16K min. Disk only

1122 GR.0 SCREENDUMP
by Jeff Davies – Llandello
List the screen to the 1122 Thermal Printer
Runs in 16K Cassette or Disk min.

Education

INTERFERENCE WAVES
by David Cheung – London
Demonstration of the interference pattern produced when waves meet.
Runs in 16K Cassette or 32K Disk min.

SAMSPELL
by Phil Brown – Newquay
Spelling and pronunciation program for youngsters SAM required.
Runs in 48K min. Disk only

SINE/COSINE CURVES
by Steven Pearsall – Birmingham.
Two programs one showing the sine/cosine of X, the other showing the variable X.
Runs in 16K Cassette or 32K Disk min

Music

SIX TUNES
by Grahame Fairall – Oxon
6 Atari Music Composer Cartridge tunes Sweet Georgia Brown, From Russia with Love, Skaters Waltz, When I'm 64 If I Could Teach the World, Onedin Line.
Runs in 16K Cassette or Disk min.

PUFF THE MAGIC DRAGON
by Grahame Fairall – Oxon
Well known tune from Basic
Runs in 16K Cassette or Disk min

CRACKING THE CODE

by Keith Mayhew Part Five

Having covered a lot of basic ground work, the previous articles should be used as reference material Binary and hex numbers were introduced, as well as some binary arithmetic, such as the addition of two numbers. The rest concentrated on the programmer's model of the computer - consisting of external memory and a central processor All the processor's instruction codes were described, showing how they affect the registers, flags, and memory, as well as describing how the instructions and data are stored in the memory and their execution.

We are now at the point where we can start the more practical side of machine code programming using an assembler and running some simple examplas It is recommended that you experiment with such simple programs Do not be disheartened if things do not go as expected, you will learn a lot from your mistakes such as saving a copy of your program in case it crashes! Lastly if you have any difficulties which you can t solve over a cup of coffee and a few back issues then I will be pleased if you drop me a line explaining your problem – no matter how simple Suggestions as to what topics you would like to see covered in future articles would be useful as well as any comments you may have

Why an Assembler?

We already know that each instruction has been given a standard three letter mnemonic to help remember its purpose In the last issue there was a complete list of these mnemonics and their associated op-codes. The term 'hand assembly' is given to the process of manually converting an assembly language program into machine code by the use of such a table However the task is made more tedious by the fact that most mnemonics have more than one op-code entry in the table, each allowing for the different addressing modes the instruction can be used in Then there is the added complication of changing all numbers into one base, say decimal for entry into the computer There are many more reasons why hand assembly becomes difficult and slow that's apart from any mistakes you can easily make when working with pages of numbers

Assemblers overcome all the problems associated with hand assembly because the process of converting assembly language (of mnemonics labels etc.) into machine code becomes just one command Assemblers have many more adventages than just converting assembly language that make the development of a program a lot easier Most assemblers are supplied with a powerful editor to create and modify assembly language programs and a 'debug or monitor program that allows you to look at memory locations, dis assemble machine code back into mnemonics and many more useful features that will help you to test and find bugs in your programs Even when writing small subroutines you will find an assembler is a luxury over doing things by hand. I would advise you to invest in one of the several assemblers available If you intend to get to grips with the code in the long run it will save you time and frustration!

This series will be centred around the ATARI Assembler Editor cartridge which is perhaps the most popular among users, it is not the best, but it is easy to use and quite comprehensive. Apart from the actual speed of assembly, other assemblers tend only to differ in minor respects or offer features which only an advanced programmer may find useful such as macros An excellent compromise over a commercial assembler is USERCOMP, although a little limited in that most of the features associated with an assembler are missing, you can enter all the mnemonics and is great for writing small routines or for experimenting with For just one pound, or for free under the exchange offer, you can't complain!

Get those Fields in File!

There are two types of files which are associated with an assembler, the source file and the object file, these reside on either disk or cassette. The source file represents the assembly language program which you create and modify with an editor This is analogous to how you enter and save a BASIC program except that the program is usually saved as pure text, i.e exactly as you typed it.

When you decide you want the machine code for the assembly language you command the assembler to compile the source file The source file will be read and an object file created, this will be actual machine code on a disk system you could binary load this file (directly from DOS) into memory Remember that any changes you want to make to your program should be made to the source file re-compiling will then give the updated version of the machine code In the object file, ready for execution At this point the fundamental difference between a language like BASIC and an assembler should be clear BASIC is an interpretive language and executes statements directly from a source file line by line and therefore BASIC has to be resident to run the program An assembler is a compiling language and works with two files – source and object – and converts all the statements in the source file into directly executable machine code in the object file, therefore the assembler does NOT have to be resident to run the object code

We will now look at the structure of the source file in some detail The file consists of linas of text each line is thought to consist of fields of characters Here a field consists of one or more adjacent characters each field is separated by one or more spaces – analogous to the words in a sentence The first field is normally a line number this hes NO effect on the assembler what-so-ever

Line numbers are used by most editors to keep track of lines and as such are completely ignored by an assembler if they are present. We shall therefore call the first field either the one at the start of the line or the one following the line number (by one space), depending on which type of editor you are using A line can have from one to four distinct fields

The Label Field

A label must start with a letter and can have any mixture of letters and digits following it forming a 'name' for the label – similar to a variable name in BASIC except there is usually a limit of upto six characters only Labels are used to represent numeric values, once a value has been assigned to a label then where ever the label is used its value will be substituted during the assembly of the program The first field on a line is reserved for a label

name – you will see the use of labels soon

The Operation Field

The second field is the operation field and is where the familiar op-code mnemonics are written, such as LDA Pseudo-operations, also known as assembler directives, are also written in this field and are used to control the operation of the assembler when compiling your source code

The Operand Field

Operands are written in the third field and are the data for what is in the operation field. The type of operation will determine if an operand is necessary or not. Any label which was previously defined (by placing its name in the first field) can be used in the operation field – the effect is the same as writing the actual value of the label in place of it so labels are effectively substitutes for fixed values, or constants.

The Comment Field

Comments are simply a string of characters commenting the operation of the program similar to the REM statement in BASIC. The start of a comment is usually signified by a semicolon (;), when this is reached the rest of the line is completely ignored. The comment field is an exception in that it can start at any place on a line. They can take up a whole line or can be placed at the end of any existing line. The use of comments helps describe what a program is doing and should be used quite liberally throughout any program as documentation. If you find an old program at the bottom of a drawer then the comments will prove invaluable to remembering what it did and how!

Neat Fields

At least one space must be used to skip a field, if the tab key is used to space fields out you will not have to worry about the exact spacing and as a bonus you will find that they line up neatly from line to line. The following shows some possible field layouts:

```
100 LABEL  LDA #$3A  Minimum spacing
200        PLA       Operation only
300 PLA              ,Same, but with
                     minimun spacing.
```

From now on tabs will be used to space these fields out. Listing 1 shows a complete program. The program loads the accumulator with the value zero, stores this at location 3000 hex and then returns to the calling program. The third line down contains – '*= $0600 the '*= is an assembler directive which sets the origin or location counter of the assembler to 600 hex. This causes the machine code to be generated from location 600 hex onwards. Such a command must preceed any program to determine where the code will be stored and can be used anywhere else, thus spliting the code up into different sections of memory if desired.

Listing 2 is the output from the assembler during the assembly. It is an exact copy of Listing 1 except that two extra columns of numbers have been appended to the left hand side. These show the contents of the location counter and the contents of the memory respectively. From this we can see that 'A9', the op-code for 'LDA' is stored at location 600 hex and that '00' the operand, or data, is stored in the next location of 601 hex, the next line shows the count or address at the next location of 602 hex as expected. Therefore, by looking at the second column of numbers we know the exact contents of the object file, i.e. the actual machine code for our program and the address of any particular byte

```
0100 ;Simple program showing the effect of
0110 ;using the tab key to space out fields.
0120         *=      $0600   ;Start code at 600 hex.
0130         LDA     #0      ;Zero location
0140         STA     $3000   ;3000 hex.
0150         RTS             ;Return to caller.
```

Listing 1

```
                0100 ;Simple program showing the effect of
                0110 ;using the tab key to space out fields.
0000            0120         *=      $0600   ;Start code at 600 hex.
0600 A900       0130         LDA     #0      ;Zero location
0602 8D0030     0140         STA     $3000   ;3000 hex.
0605 60         0150         RTS             ;Return to caller.
```

Listing 2

```
0100 ;Program to show the use of labels to
0110 ;represent memory locations and data.
0120 SUM    =       $4000   ;Address where sum is stored.
0130 INCR   =       $13     ;Increment value.
0140        *=      $0600   ;Set location counter.
0150        CLC             ;Clear carry for addition.
0160        LDA     SUM     ;Get current sum.
0170        ADC     #INCR   ;Add on the increment.
0180        STA     SUM     ;Store the result back.
0190        RTS             ;Return.
```

Listing 3

Label It

Listing 1 does not have anything in the first field i.e. labels. If a label is placed in the first field then it is assigned a value. This value will be the contents of the location counter, i.e. the current address. Another method exists for assigning a value to a label by using the '=' or equate directive in the second field, the operand following this will then be the value assigned to the label. It should be noted that the assignment of a value to a label can only be made ONCE during a program. Once given a value, the label can then be used in the operand field as the data to whatever is in the previous field. When assembled the value of the label is substituted for the label's name.

Labels are used to give names to memory locations or data. One advantage of using labels is that they aid your memory, instead of remembering or looking up a value repeatedly, a meaningful name can be given to the value and then the label name can be used where ever needed. Another advantage is obvious: if you want to change a value then you haven't got to search the entire program changing the value many times instead you would only need to change the value of the label (at the start of the listing).

Listing 3 shows the use of labels to represent memory locations and data. The program adds an increment to a memory location and stores it back again. Line 120 assigns the value of 4000 hex to SUM, this is the address where the number to be incremented is stored. The next line assigns the value of 13 hex to INCR, this represents the amount to be added to the sum. The rest of the program is as normal except that where an address or data is needed the label name is used instead. When assembled, the label names are simply substituted for their values in the operand field. Therefore the effect is exactly the same as if, for example, line 160 read "LDA $4000".

If we now wanted to change the address of SUM we only have to change it

```
0100 ;Program showing the use of labels
0110 ;to 'mark points' in programs.
0120 COUNT   =     $4000   ;Holds current count.
0130         *=    $0600   ;Set location counter.
0140 ADD     CLC           ;Clear carry for addition.
0150         LDA   COUNT   ;Get count.
0160         ADC   #2      ;Add on increment.
0170         STA   COUNT   ;Store new value back.
0180         JMP   ADD     ;Jump back to start.
```

Listing 4

```
0100 ;Program to find the 'value' of
0110 ;an ASCII number.
0120 CHAR    =     $4000   ;Holds ASCII character.
0130 NUM     =     $4001   ;Holds value of the number.
0140         *=    $0600
0150         SEC           ;Set carry for subtraction.
0160         LDA   CHAR    ;Get ASCII character.
0170         SBC   #'0     ;Subtract value of ASCII 0.
0180         STA   NUM     ;Save as a number.
0190         RTS           ;Return.
```

Listing 5

on line 120 as opposed to having to change both line 160 & 180 if we had written the actual value of $4000 there Listing 4 uses two labels, the first is defined in the same way as before and is the address of a count. Line 140 has the second label, ADD however this time it preceeds e 6502 mnemonic. In this case the value assigned to ADD is the address of the instruction it preceeds, which happens to be the start of the program, i e 600 hex

Line 180 is a jump instruction, it jumps to the address held in ADD so in this case it is the same as writing

JMP $600' Thus the program continues in e never ending loop adding two to COUNT every time it goes around the loop Labels can, of course, be placed anywhere in a program, a jump or branch can then be made to the label, instead of trying to work out which location the instruction is stored in. The use of labels in this way can be thought of as 'marking points' in a program which can then be jumped to by reference to the label or 'marker'

Odd Expressions

Yet another advantage of an essembler is to evaluate expressions in the operand field For instance

LABEL = 2*3+1 will evaluate the expression on the right hand side to be seven and then assign the value seven to LABEL. Such an expression can contain other label names which are already defined these expressions can also be used as an operand to a 6502 instruction This facility does not provide e short cut to doing multiplicaton and division in machine code expressions are evaluated once at assembly time and are then used in programs as a fixed value — just as e label has a fixed value The use of these expressions does however, seve you having to use a calculator to work out a constant, instead the computer works it out for you et essembly time A useful expression is the single quote followed by an ASCII character, when the expression is evaluated the ASCII value of the character becomes the operand — this saves looking up characters in tables of ASCII values and writing that value down Listing 5 demonstrates the use of this Assuming the location et CHAR contains the ASCII value of a digit between 0 and 9, the program will place the actual number that character represents in location NUM The ASCII value of 0 is 48 decimal, the characters 1 to 9 have the consecutive values of 49 to 57, so to convert the character to the number it represents we simply have to subtract the value of ASCII 0 from it. Line 170 does the subtraction, the 0 is evaluated to be 48 the ASCII for the character 0, the result is then stored in location NUM For instance, if the ASCII character was 9 then 57 would be in location CHAR, by subtracting 48 from this we get the actual value of 9 stored in location NUM.

Getting It Together

Having seen e lot of examples on the features of an assembler, we will end by writing a program which when called by BASIC, will multiply two integer numbers together and return the answer to BASIC Although not being spectacular, it will hopefully bring together a lot of ideas and principles covered so far in this series and should be fully understood.

Using the User

Before we can start we need to look at the mechanism by which BASIC ellows the user to call mechine code routines — by the aptly named USR' command. This command also allows numbers, known as parameters, to be passed back and forth between the two The general form of the command is

X=USR(ADDR PAR1 PAR2 ...)

Where 'ADDR' is the address in memory where the routine starts and 'PAR1' 'PAR2 etc.. ere the parameters to be passed, the parameters are separated by commas and the number of these can vary from none to as many as BASIC will let you type on one line All these numbers should be integers between 0 and 65535 (although BASIC does round-off any non-integers) An integer, in the same range can also be passed back into the variable X of the USR command

The numbers BASIC passes ara saved on the stack. By doing this we don't have to worry about exactly where they are stored in memory, to retrieve these numbers in the machine code we use the PLA commend which takes the last entry of the stack and puts it into the accumulator The first number pulled back tells us how many parameters BASIC stored on the stack e g. if zero then it tells us thet there were no parameters supplied by the user This can be used to see if the actual number of parameters is equal to the number you expected, if not, the error can be dealt with Most of the time it is ignored and assumed that the user knows how many parameters he should supply, however using this method means that if you do give the wrong number of parameters it will almost definitely crash on you, so be careful Once the parameter count has been taken off the stack, the parameters are then removed, two bytes for each The first PLA will give the high byte of the first parameter in the USR command the second PLA will return the low part of this same parameter the rest are taken off in the same manner until alf are removed. If everything went well, then the return address will be left on the top of the stack ready for an RTS to take you back into BASIC To return a value to the BASIC variable used in the USR command then you simply store the low part in location 212 end the high part in 213 (decimal) On return BASIC sets the variable equal to the value of this two byte integer

Doing the Multiplication

The easy wey to multiply two numbers together is simply to add one of them repeatedly to a result, the number of times will be determined by the other number for instance 2*3 is the same as 2+2+2, i e. 2 added on three times This is obviously very inefficient, especially for large numbers as a lot of addition will be involved better methods do exist but we will stick to this one for its simplicity Using our method, we need to add the multiplicand to the result by the number of times indicated by the multiplier For the example of 2*3, the multiplier is 3 thus we add 2 onto the result 3 times to get the answer of 6 However, this is exactly the same as 3*2, now the multiplicand is 3 and the multiplier is 2 this means that we do one less eddition then before, i.e 3+3 This is better illustrated by 1*255 which takes 255 additions of 1 instead of 1 addition of 255 for 255*1 The method, or algorithm, for our multiplication now becomes Firstly if the multiplier is greater than the multiplicand then swap them over, we then zero the result and add the multiplicand to the result by the number of times as the value of the multiplier

```
                0100 ;Simple multiplication of two single byte
                0110 ;integers by repeated addition.
00CB            0120 MLTPND  =     $CB      ;Multiplicand.
00D4            0130 RESULT  =     212      ;Result for BASIC.
0600            0140         *=    $0600    ;Start on page 6.
0600 68         0150         PLA            ;Discard number of data.
0601 68         0160         PLA            ;Discard high byte.
0602 68         0170         PLA            ;Get low byte.
0603 85CB       0180         STA   MLTPND   ;Save as multiplicand.
0605 68         0190         PLA            ;Discard high byte.
0606 68         0200         PLA            ;Get low byte.
0607 AA         0210         TAX            ;Save as multiplier.
0608 C5CB       0220         CMP   MLTPND   ;Compare both.
060A 9005       0230         BCC   NOSWAP   ;OK if MLTPLR<MLTPND.
060C A5CB       0240         LDA   MLTPND   ;Swap MLTPLR
060E 86CB       0250         STX   MLTPND   ;for MLTPND
0610 AA         0260         TAX            ;so that MLTPLR<MLTPND.
0611 A900       0270 NOSWAP  LDA   #0       ;Zero the result
0613 85D4       0280         STA   RESULT   ;low
0615 85D5       0290         STA   RESULT+1;and high.
0617 E000       0300 MLTPLY  CPX   #0       ;If MLTPLR=0 then
0619 F00F       0310         BEQ   EXIT     ;exit loop.
061B 18         0320         CLC            ;Clear carry.
061C A5D4       0330         LDA   RESULT   ;Add in MLTPND
061E 65CB       0340         ADC   MLTPND   ;to the result
0620 85D4       0350         STA   RESULT   ;and save back.
0622 9002       0360         BCC   NOCARY   ;No carry so branch.
0624 E6D5       0370         INC   RESULT+1;Else adjust high byte.
0626 CA         0380 NOCARY  DEX            ;Decrement count
0627 4C1706     0390         JMP   MLTPLY   ;and go back.
062A 60         0400 EXIT    RTS            ;Exit back to BASIC.
```

Listing 6

Listing 6 shows the completed assembly program the following describes how you might go about entering it into an assembler to compile it. If you have an assembler editor cartridge then you would type in everthing except the two columns on the far left, the comments are, of course, optional. If you have a different assembler then you would need to change a few minor things see the appropriate manual for some help Table 1 gives some guidelines for conversion between most of the popular assemblers. Before assembling the code you would save a copy of this as a source file (disk owners should note that the extension of SRC is often used to denote the source file), assembly of the program is then done to an object file (for disk the extension of OBJ is widely used) Assuming there were no errors then the contents of the object file would be the same as the second column of Listing 6 and would load into the addresses shown in the first column, i.e. page 6 (600 hex) onwards

How it Works

The equates on lines 120 & 130 set up two labels, MLTPND will hold the multiplicand and is at location CB hex, RESULT is set to point to location 212, this is where BASIC picks up the value for the variable in the USR statement. Next the location counter is set so that the code produced is placed from 600 hex onwards The first PLA gets the number of BASIC's parameters into the accumulator however, this is discarded, as is the high byte of the first parameter, this leaves us with the lower half in the accumulator after the third PLA and is saved as the multiplicand by the following STA instruction at location MLTPND, i.e CB hex. Two more PLA s leave the low part of the second parameter in the accumulator and is saved in the X register by the TAX instruction as the multiplier Only the low part of each parameter is saved so that neither number can exceed 255, by doing this the result will not need more than two bytes which is the maximum size we can return to BASIC's variable.

We now want to see if the multiplier is less than the multiplicand to minimise the number of additions A copy of the multiplier is still in the accumulator at this point, the compare instruction then subtracts the multiplicand from it leaving only the N Z and C flags affected We want to skip the swap over if the accumulator was less than the data, i.e the comparison left a borrow condition because the larger data was subtracted from the smaller value in the accumulator If a borrow occured then the carry (C flag) would be cleared, the next instruction, BCC, would then branch over the following three instructions to NOSWAP, thus not doing the swap You can see by looking at the machine code on the left, that the assembler has converted the address of NOSWAP (611 hex) into the offset of 05, following the opcode of 90 for the BCC instruction, yet another thing the assembler does for you!

If the branch failed then the instructions on lines 240, 250 & 260 swap the multiplier and the multiplicand over the accumulator is loaded with the multiplicand and then the multiplier in the X register is saved as the new multiplicand finally the accumulator is moved back into the X register as the new multiplier Which ever path was taken, when the LDA instruction is reached at NOSWAP, the multiplier will be in the X register and the multiplicand will be in MLTPND with X being the lesser of the two At NOSWAP the accumulator is loaded with zero and is stored at location RESULT (212) as the result will take two bytes we must also zero the high part at the next location. Rather than define another label to point to location 213 RESULT+1 is used, this is equivalent to 212+1 which is location 213 the high part of the result.

The multiplier is in the X register so that it can be used as a count, every time an addition is done we will take one from the count until it reaches zero The test for this count being equal to zero is made on line 300, which compares the X register to the number 0, if it is then the 'branch if

Description	Atari Assm/Edtr Cartridge	Synapse Syn Assm Disk	O S S Mac 65/Bug 65 Disk or Cartridge	Atari Macro Assm Disk
Set origin (Location counter)	*=	.OR	* =	ORG
Value of location counter	*	*	*	* 0
Equate (assignment)	=	.EQ	=	= or EQU
Define bytes/ characters in memory	.BYTE	.AT .HS .AS	.BYTE .SBYTE .DBYTE	DB DC
Define words in memory	.WORD	.DA	.WORD	DW
Skip bytes	*=*+	.BS	*=*+	DS

Table 1 Assembler Conversion Chart

Continued on page 32.

FAST FILL

BY KEITH MAYHEW

So you thought you had seen the last of Graphics 8 utilities with the excellent Quick Plot, eh? Well here is yet another! If you have ever used the fill routine in the operating system, via the XIO command, you would have realised its limitations - it only fills four sided shapes and is very slow. Drawing utilities, such as ATARI ARTIST, can fill in any shape you design and quite fast. Fast Fill uses a similar method to these programs and allows you to fill any shape on a Graphics 8 screen, and fast.

Listing 1 is a simple demo program in BASIC. Type it in and remember to save a copy before you run it. If the program crashes then re-load it and check all the numbers in the DATA statements. Assuming the program runs OK, the screen should clear after a short delay while the machine code is read in. The little flashing pixel in the middle is a cursor! It can be moved in all eight directions with a joystick and lines can be drawn by pressing the fire button down. Once you have drawn a shape place the cursor inside it and press the START button to fill it up. Remember that if you do not properly enclose the area to be filled the whole screen will be filled up, even if only one pixel was missing.

To use the machine code fill routine in your own programs list lines 30000 to 32380 to disk or cassette (e.g. LIST "C:",30000,32380) and then enter it onto your own program (e.g. ENTER "C:").

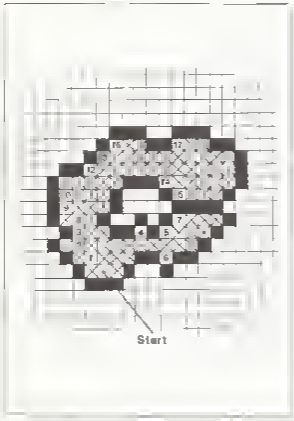
To initialise the machine code you will have to place a GOSUB 30000 at the start of your program and then call the machine code by something similar to Z=USR(ADR(FILL$),X,Y), where the pixel at which to start the fill is held in X and Y. A word of caution - in order to make the fill fast there is no error checking to see if it goes off the top or bottom of the screen - CRASH! To avoid this problem either make sure the shape is completely contained or draw a border around the screen as in the demo program.

The fill routine has been split into two parts - the initialisation section is held in a BASIC string (FILL$) and the main part is in page 6 (no - not the magazine!). The initialisation section first points a software stack to just under the screen memory, this grows downwards as pixel addresses are pushed onto it. The rest of this section converts the X & Y co-ordinates into a screen memory address for that pixel. As there are eight pixels per byte in this mode a 'plot mask' is calculated which contains one bit set to a '1' - so that when ORed onto the screen at the screen address - will plot the correct pixel. A jump is then made into the main section in page 6.

The method by which the shape is filled is in horizontal runs or lines. While a run is in process the adjacent lines are examined and continuation addresses are

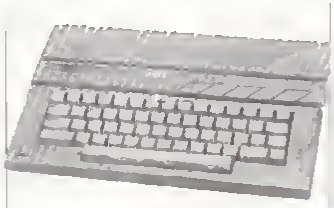
Listing 1

```
EI 1 REM ********************************
NH 2 REM
BG 3 REM FAST FILL
NJ 4 REM
MJ 5 REM by Keith Mayhew
NL 6 REM
VH 7 REM for Monitor Magazine
NN 8 REM
EQ 9 REM ********************************
GR 100 GOSUB 30000
LJ 110 GRAPHICS 8+16:X=160:Y=96:POKE 709,
   15
KZ 115 COLOR 1:PLOT 0,0:DRAWTO 319,0:DRAW
   TO 319,191:DRAWTO 0,191:DRAWTO 0,0
FG 120 S=STICK(0):LOCATE X,Y,C:COLOR 1-C:
   PLOT X,Y
ED 130 FOR I=0 TO 5:NEXT I:COLOR C:PLOT X
   ,Y
CU 140 X=X+(S=5 OR S=6 OR S=7)
AP 150 X=X-(S=9 OR S=10 OR S=11)
CO 160 Y=Y+(S=5 OR S=9 OR S=13)
DB 170 Y=Y-(S=6 OR S=10 OR S=14)
ML 180 IF  NOT STRIG(0) THEN COLOR 1:PLOT
    X,Y
AC 190 IF PEEK(53279)=6 THEN Z=USR(ADR(FI
   LL$),X,Y)
MD 200 GOTO 120
NN 30000 REM Read in machine code for fil
   l.
ZD 30010 DIM FILL$(116)
YF 30020 FOR I=1 TO 116:READ D
EM 30030 FILL$(I,I)=CHR$(D):NEXT I
AB 30040 FOR I=0 TO 164:READ D
DC 30050 POKE 1536+I,D:NEXT I
DX 30060 RETURN
HH 32000 DATA 104,173,229,2,133,206,173,2
   30
MS 32010 DATA 2,133,207,104,141,186,6,104
SF 32020 DATA 141,185,6,169,0,141,189,6
FF 32030 DATA 104,104,10,46,189,6,10,46
LQ 32040 DATA 189,6,10,46,189,6,133,203
HL 32050 DATA 174,189,6,134,204,10,46,189
IY 32060 DATA 6,10,46,189,6,24,101,203
VR 32070 DATA 133,203,165,204,109,189,6,1
   33
MF 32080 DATA 204,173,185,6,41,7,170,169
JX 32090 DATA 0,56,106,202,16,252,133,205
NI 32100 DATA 173,186,6,74,173,185,6,106
IF 32110 DATA 74,74,24,101,203,144,2,230
ZR 32120 DATA 204,56,233,40,176,2,198,204
SQ 32130 DATA 24,101,88,133,203,165,89,10
   1
FT 32140 DATA 204,133,204,76,0,6
HQ 32150 DATA 160,40,177,203,36,205,208,2
   0
UY 32160 DATA 6,205,144,240,30,205,165,20
   3
ZL 32170 DATA 208,2,198,204,198,203,177,2
   03
VR 32180 DATA 208,234,240,242,169,1,141,1
   87
RR 32190 DATA 6,141,186,6,76,205,144,0
PR 32200 DATA 192,205,230,203,208,2,230,2
   04
UN 32210 DATA 168,40,177,203,36,205,208,5
   6
NW 32220 DATA 5,205,145,203,173,187,6,240
DX 32230 DATA 11,160,0,177,203,36,205,208
KD 32240 DATA 3,32,146,6,160,0,177,203
GY 32250 DATA 141,187,6,173,186,6,240,14
XL 32260 DATA 160,80,177,203,36,205,141,1
   88
AP 32270 DATA 6,208,3,32,146,6,160,80
RX 32280 DATA 177,203,141,189,6,76,36,6
UB 32290 DATA 165,207,205,230,2,208,7,165
BD 32300 DATA 206,205,229,2,240,56,160,0
KK 32310 DATA 162,2,177,206,149,203,230,2
   06
GJ 32320 DATA 208,2,230,207,202,16,243,76
CX 32330 DATA 0,6,166,204,56,165,206,233
BR 32340 DATA 3,133,206,176,2,198,207,192
HU 32350 DATA 56,233,40,176,1,202,24,101
BG 32360 DATA 203,160,2,145,206,138,105,0
VA 32370 DATA 136,145,206,165,205,136,145
   ,206
OE 32380 DATA 96
```



Diagram

placed onto a stack, when a run is complete the last stacked address is removed and the process continues from there until the shape is filled i e the stack is empty When first entered the routine searches to the left of the current line until a border or edge is reached. The line is then plotted pixel by pixel to the right until the right border of the run is reached. While doing this it looks at the two



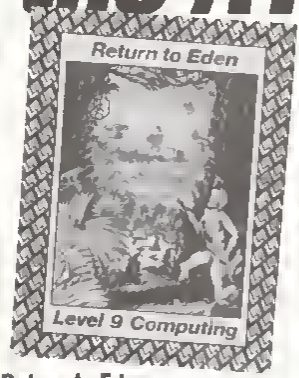
adjacent lines to see if a left edge is detected, this requires knowing the state of the previous pixel as well as the current one. To achieve this two flags are kept updated with the state of the last pixel examined If the previous pixel (flag) was set and the current pixel isn't then the pixel address and plot mask are stacked If the extreme left edge of the adjacent lines fall before the current one then this starting edge would not be detected, it is for this reason that both flags are set at the start of a line Then if the first pixel directly above (or below) the start of the current line is cleared then the point will be stacked. If the stack is then empty it is the end of the fill, else the last address is pulled back and jumps back to the start which continues the same process from this point. To help you understand the fill process the diagram shows a filled shape The numbers indicate the order and the place where left edges or

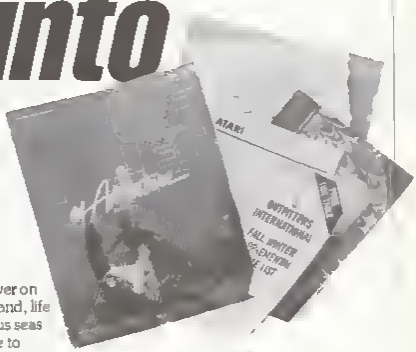
Point being used for PLOT	Stack contents After PLOT
START	1
1	2
2	3,4,5,6
6	3,4,5
5	3,4,7
7	3,4
4	3
3	8
8	9
9	10
10	11
11	12
12	13,14
14	13,15
15	13
13	16,17
17	16
16	EMPTY

Table

starting points were found and stacked The table then shows the order in which the lines where drawn from these stacked values and shows the pixel numbers on the stack after the line has been drawn.

Adventure into the ATARI





by STEVEN HILLEN

Return to Eden

32K Cassette £9.95

This is Level 9's long awaited sequel to Snowball which has become a bit of a classic because of its enormous size For those of you unfamiliar with Snowball s plot, here it is You ara Kim Kimberly, appointed to watch over the safety of two million colonists on board the Snowball 9 spaceship on route to Eden another planet. You awaken to find that the ship is headed towards a star and all the robots are now homicidal! At the end of the adventure you confront and defeat the saboteur, and rescue the Snowball

In the sequel false video-tapes show that Kim him (her?) self was tha saboteur, and Kim is sentenced to death Luckily, Kim escapes in a strato-glider and lands on Eden Unfortunately the colonists are out for revenge and turn the ship s rockets on you devastating the land if you should survive this, you emerge in a strange jungle where most of the normally friendly wildlife is after your blood! On the subject of death you are at least given a few chances, appearing strangely resuscitated inside a tree-pod

Through a 'see-bee' telescope you can preview the task ahead with minefields, gunships, towers and the jungle all barring your way to the city Enoch, which you must rescue from the onslaught of the jungle and where you must prove your innocence to the colonists You can expect this to take quite some time

Well, you can t ask for a more dramatic plot then this and coupled with a good parser this game deserves to be a hit. I can t wait to see the final part of the trilogy, 'The Worm in Paradise' where you are rumoured to have to try and discover whether the whole universe is just some puppeteer's toy

Cutthroats

48K Disk £32.50

As an out of work deep sea diver on the miserably dull Hardscrabble Island, life is incredibly boring. The treacherous seas surrounding the island defy anyone to retrieve the sunken treasures they conceal. Then, one night, an old seamate of yours, Hevlin, arrives at your hotel room door He passes you a book revealing the location of a large haul of treasure, then dashes off You hear a scuffle outside as Hevlin is murdered. You awake the following morning to find a mysterious note that has baen slipped under your door, inviting you to a meeting.

If you ve ever wanted to play the leading role in a pirate adventure, then play Cutthroats' No other game I ve played has quite had such completely convincing characters with such realistic responses The plot also twists and turns, sweeping you into the story

In order to claim your riches, you must join forces with the likes of Johnny Red and his band of cut throats Even when you do gain their confidence, they cannot be trusted, for the Weasel is ready to slit your throat at the slightest hint of treachery

To start the game you are supplied with a rather useful map locating the shipwrecks a price list for Outfitters International who can supply your diving gear, and a tide timetable. Each time you play, you may have to search for a different wreck as there ara 4 in the game There seems to ba no limit to what you can do in this adventure for it understands over 800 words I keep going back to this game it's just like being in a Robart Louis Stevenson novel, not knowing what will happen next Infocom are still showing the rest how adventures should ba written

Emerald Isle

32K Cassette £6.95

This is an adventure that seems to have something of everything, and is again up to Level 9's consistently high standard It is being sold for £3.00 less than usual as it is rated as an easier adventure although I still think it's pretty challenging. A line-map of the island is supplied in an ever-improving style of packaging along with some large glossy posters

You arrive on Emerald Isle (somewhere in the Bermuda Triangle) by silk perachute which unfortunately snags in the trees Vultures watch your attempts to escape Next you discover a tree-top fantasy city which is strangely deserted except for a bored King who seems to have issued some kind of challenge If you re carrying the right goods, then both the guard and seamstress will help you

Back down on the ground, avoid climbing onto the railway tracks, otherwise the inevitable happens Instead you can enjoy a ride to the seaside station which backs onto a glorious beach It soon becomes apperent that you need some light to explore the mine and the cave (not the inside of the spider!), so if you can build a canoe, the second island may provide some help

In this game, points are scored for swelling the King s coffers although the ultimate objective is to escape from the Isle

Why then Is it rated as a baginner s adventure? Well unlike all the other Level 9 adventures there is usually just one way you can go to proceed with the game Instead of deciding where and when to go, you need only find a method of getting there

Emerald Isle is unusually good value for money and great fun to play as there is so much variety Thanks to Level 9 for producing such games and for sending me review copies

Spiderman

16K Cassette £7 95, 48K Disk £17.95

This is the second of Scott Adams 'Questprobe' series, the first being 'The Incredible Hulk' of course The game is available on disk or cassette but only the disk version has graphics

Spiderman starts in a mysterious office block virtually surrounded by Marvel characters As with the Hulk, your score is increased by gathering the gems together However, the Hulk was at least consistant and vaguely credible, whereas Spiderman fails on both counts. The reason for these gripes are as follows The whole situation seems very contrived for in almost every room there is another trumped-up' superhero who must be defeated or used in some way or another OK, Spiderman himself may not be totally convincing but the Sandman and Madame Web are absolutely comical Unfortunately there are also some logical faults within the game, an example being when going down one floor, out of the window and finding yourself at the very top of the building

The screen format and parser have been left unchanged from previous adventures so they are adequate but slow at times. There seems to be a number of themes that may run through the whole Questprobe series One is the method of resuscitation when you die another is the re-appearance of the natter energy egg I hope that the third in the series will not be another treasure hunt, one or two of the win/lose situations such as The Count would be a welcome change All in all then, Spiderman is to me at least, just another unmemorable adventure

Dallas Quest

48K Disk £14 95

Well here is the first adventure that Datasoft have written since the graphically excellent and challenging 'Sands of Egypt Not being a Dallas freak myself South Fork Ranch isn t as exciting a scenario as ancient Egypt. The game loads with a picture of J R and the theme music from the TV series and the game starts in the living room with Sue Ellen The graphics are again quite outstanding, but the adventure itself is poor, simple and illogical It has been completed by some adventurers in less than an hour, hardly giving them their monies worth nor much satisfaction

The game centres on finding a new oil deposit near the ranch, which is not too difficult if you can avoid being beaten up at the main gate by some local yobbos or by the equally improbable giant rat which lurks in the barn Just as inconsistent is the monkey blocking the hole in the boat with his tail!

It really is a shame to see such excellent graphics sound and overall programming wasted on such an intrinsically poor game In short then, recommended for incurable Dallas fans only

Letters Section

Well, this time you seem to be running out of questions for me to try and answer Maybe you re all such experts by now that you no longer need assistance? However I haven t heard of anyone who's yet escaped from Issue 8's 'Nightmare Reflections' so here are some gratuitous clues which you should follow very closely

Push around, ways abound.
Touch to show the node the watch completes the code
It s hard to reside in a SQUARE of SIDEs.
Rainbow colours one to seven just add to the problem
Initialise the paper clue for the final thing to do
Nine more true moves make, and then you'll be awake
Fishy draughts are just for laughs.
One false step then, and it'll be another ten!

Back to commercial adventures again, the remaining unanswered questions are below If you can answer any of them I'd be pleased to hear from you If you re hopelessly stuck with a game, then write in, and I ll try to help If you ve completed an adventure, again please let me know and I'll add it to a list I'm compiling for next time Thanks to all those who wrote in this time.

Lastly, thanks to Peter Lister for answering several queries, and for letting me preview his rather good adventure The Amulet

Unanswered Questions

Savage Island II
Can't kill dinosaurs nor change back from Neanderthal

Escape from Pulsar 7
Can t remove cups from captain's cabin ceiling

Waxworks
Where is the well?

Curse of Crowley Manor
Have combination but can t get past monster in numerical lock' room

Answered Questions

Ten Little Indians
Can t do anything with couch nor sofe

```
10 DIMA$(40) A$="MY_MR*S}* \
ON*RO\ \ SXQ8*NSKV* CB=-"
20 FOR A=1 TO LEN(A$) A$(A,A)=
CHR$(ASC(A$(A,A))-10) NEXT A.? A$
```

by Tony Cheung

BOOTBASE

by Sol Negrine

This program reads the directory of your 'Multiboot' disks (sectors 48 and 49) and optionally allows you to print out labels to show program names, sizes and free space on each disk, and also it appends the entries to a DOS file named PROGLIB.DB which is created and/or unlocked as required. This file is compatible with the APX Diskette Librarian system, which will allow you to sort and print your entries, (note that any existing data on the DOS file is retained).

You may wish to customise the label formats to suit your own printer, this can be done by ammending the PRINT statements in lines 385 – 414 as appropriate. The use of PRINT #3 statements instead of LPRINT is to allow one to compile BOOTBASE with the excellent ABC Compiler from Monarch Data Systems. Please note that the program features delay loops which will be excessive if you RUN the program from BASIC without compilation. Ammend these as required. The delay loop is called via WAIT=2500:GOSUB 1100 in lines 200,800,1010 and 1630 (just change the value assigned to WAIT.

If you do not have or do not wish to use the APX Diskette Librarian, then the output file D1:PROGLIB.DB can be treated simply as a fixed format DOS text file. You may print it directly from DOS using option C (copy file) by typing PROGLIB.DB,P, which will copy the file to your printer. You could also read it into Atariwriter (or any other text processing program), and even ammend the record layout in line 660. But how ever you use BOOTBASE, I hope you have fun!

NOTE: In this program, anything which is underlined should be entered in INVERSE!

```
RX 1 DIM NAME$(200),D$(4),SIZE$(10),R$(20
   ),B$(1),BUF1$(128),BUF2$(128)
LX 2 DIM R1$(10000),R2$(2000),R3$(500),R4
   $(500)
GN 3 GRAPHICS 0
SR 10 DATA 104,32,83,228,96
QN 20 FOR I=1536 TO 1540
YR 30 READ J:POKE I,J
IG 40 NEXT I
KY 50 POKE 752,1:POKE 739,14:POKE 710,116
   :POKE 712,116:R=0
YP 60 PRINT CHR$(125):? "    MULTIBOOT BO
   OTBASE - By Sol"
AF 70 ? "    ----------------------------"
   :?
QF 80 ? "This prograe will read the direc
   tory"
DL 90 ? "of your Multiboot disks in Drive
   1.":?
CU 100 ? "You can choose to "
FR 110 ? "          - Print labels"
JT 120 ? "          - Create a Proglib Data
   base"
QM 130 ? "          - Both of the above ."
VP 140 ? :? "Enter your choice (P,C,B)";
TG 150 POKE 764,255:OPEN #1,4,0,"K:"
IS 160 IF PEEK(764)=21 THEN ? :? :? "You
   have chosen both":C=2:GOTO 200
XI 170 IF PEEK(764)=10 THEN ? :? :? "You
   have chosen to print labels":C=1:GOTO
    200
KT 180 IF PEEK(764)=18 THER ? :? :? "You
   have chosen to create a Proglib":? "Da
   tabase":C=3:GOTO 200
OR 190 GOTO 160
AV 200 WAIT=2500:GOSUB 1100
SM 205 POS=ADR(BUF1$):PAGE=1:SEC=48
BF 206 I=INT(POS/256):J=POS-I*256
IS 210 POKE 769,1:POKE 770,82:POKE 772,J:
   POKE 773,I:POKE 778,SEC:POKE 779,0
EB 220 IF PAGE=2 THEN 340
HJ 300 ? CHR$(125):? "Plaaae put a Multib
   oot diak in D1 ":? "and press RETUNN ,
    or * to exit."
CI 310 ?
RR 320 GOSUB 1200
CA 330 ERR=0
EU 340 X=USR(1536):IF PEEK(771)<>1 THEN 1
   000
KY 341 IF PAGE=2 THEN 347
NV 342 FOR I=1 TO 128:B$=CHR$(PEEK(POS+I-
   1)):BUF1$(I,I)=B$:NEXT I
CM 346 IF PAGE=1 THEN PAGE=2:POS=ADR(BUF2
   $):SEC=49:GOTO 206
ZE 347 FOR I=1 TO 128:B$=CHR$(PEEK(POS+I-
   1)):BUF2$(I,I)=B$:NEXT I
TR 350 GOSUB 700
UI 355 IF C>2 THEN 500
IR 360 ? :? "Please ready printer ,":? "a
   nd press RETURN"
UM 370 GOSUB 900
NA 375 CLOSE #3
IS 380 TRAP 2000
NP 385 OPEN #3,8,0,"P:"
SN 390 PRINT #3
BQ 391 PRINT #3
ST 392 PRINT #3
SW 393 PRINT #3
MA 395 PRINT #3;"MULTIBOOT DISK ";D$;" SI
   DE ";S
WF 396 PRINT #3;"----------------------
   --":PRINT #3
CK 397 PRINT #3;"PROGRAM                SECTO
   RS":PRINT #3
EP 398 FOR I=1 TO N
QV 400 R$=NAME$(I*20-19,I*20)
IF 405 J=ASC(SIZE$(I,I))
SK 408 PRINT #3;N#;"  ";J
FN 410 NEXT I
HO 411 PRINT #3:PRINT #3;"FREE SECTORS ";
   665-TOT
MP 415 CLOSE #3
UO 420 IF C<2 THEN 205
DG 500 FOR I=1 TO N
TU 510 N=N+1:IF N>500 THEN 1600
US 520 R1$(N*20-19,R*20)=NAME$(I*20-19,I*
   20)
RO 530 R2$(N*4-3,N*4)=D$
XC 540 R3$(R,R)=S$
ZD 550 R4$(N,N)=SIZE$(X,X)
GN 560 NEXT I
DQ 570 GOTO 205
RI 600 IF C<2 OR N=0 THEN 1300
CN 605 ERR=0
YT 610 ? :? "Please insert a DOS 2.0 Data
   Disk":? "in D1. I shall append to a f
   ile"
CR 620 ? "called PROGLIB.DB with ";N;" re
   corda."
PV 630 ? "You can use the PROGLIB database
    on"
UJ 640 ? "it for sorting,printing etc."
RU 645 ? :? "Press RETURN when ready."
UF 650 GOSUB 900:TRAP 2500
WB 651 XIO 36,#4,8,0,"D1:PROGLIB.DB"
GH 652 CLOSE #1:OPEN #1,9,0,"D1:PROGLIB.D
   B"
DL 655 FOR I=1 TO N
XE 656 R=ASC(R4$(I,I)):D$=BTN$(N):D$(LEN(
   D$)+1,4)="   ":D$(4,4)="*"
AA 660 PRINT #1;R2$(4*I-3,4*I);N3$(I,I);"
          ";R1$(20*I-19,20*I);"  R/BO
   OT            ";D$
BS 670 NEXT I:CLOSE #1
RB 680 ? :? "FILE D1:PROGLIB.DB NOW WRITT
   EN"
LK 690 GOTO 1300
SJ 700 R=0:TOT=0:POS=ADR(BUF1$)
BQ 705 FOR I=1 TO 10:IF PEEK(POS-1+I)=0 T
   HEN 720
ZU 710 N=N+1:J=PEEK(POS-1+I):TOT=TOT+J:SI
   ZE$(I)=CHR$(J):NEXT I
YJ 720 IF TOT>665 THEN 800
FJ 730 ST=1
OY 735 FOR I=1 TO R
SO 736 FOR J=1 TO 20
BF 737 IF ST>998 THER POS=ADR(BUF2$)-128
PZ 740 B$=CHR$(PEEK(POS+29+ST))
BU 741 NAME$(ST)=B$
EN 742 ST=ST+1
FN 745 NEXT J:NEXT I
```

Continued on page 29

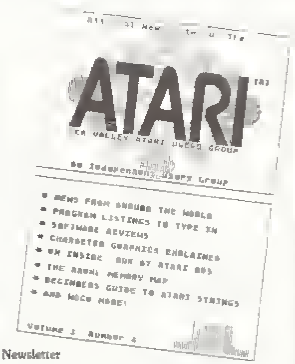
PROFILE ON LEA VALLEY

by Matthew Tydeman.

In 1980 I purchased a 16K Atari 400 for £349, (your not the only one who paid a fortune in those early deys) with a 410 cassette player end, of course, the Star Raiders cartridge the 8K wonder of Atari! Within 6 months I was totally bored with game playing and I began to dig deeper into the possibilities of my computer Finding out how it worked, and how I could achieve graphics as good as Star Raiders (great ambition eh!) Yes you guessed it, I could'ntl So, where was I to turn? Other users seemed to be my only hope, they must be having the same problems as me I thought. I rushed back to the shop where I had bought my machine end I had a long talk with the proprieter He gave me the phone number of a man who had asked the same questions only hours before me I rang, and within 2 months the 'Atari Users Group was set up, with a member list of 5, including the officials!

At the beginning, Nigel Fowler was the President end I wes made Vice-President This may seem a little odd, but in those early days (before Atari (UK)) instructions on how to form a users group came from the USA, so it is no surprise that the clubs were modelled on the Amencan Presidential system At the start we could give little help to our members except to discuss games and new releases, (at that time Caverns of Mars was considered to be great!)

Slowly we got it together and since then we have gone from strength to strength We advertised in local shops and newspapers in order to increase our



numbers We did naughty things too, like looking up owners names in shop sales lists but all in a good cause we feel. Soon, meetings in our homes became too crowded and our only choice was to hire a hall and this is whet we did We rented a small Church Hall and we still hold meetings there to this day

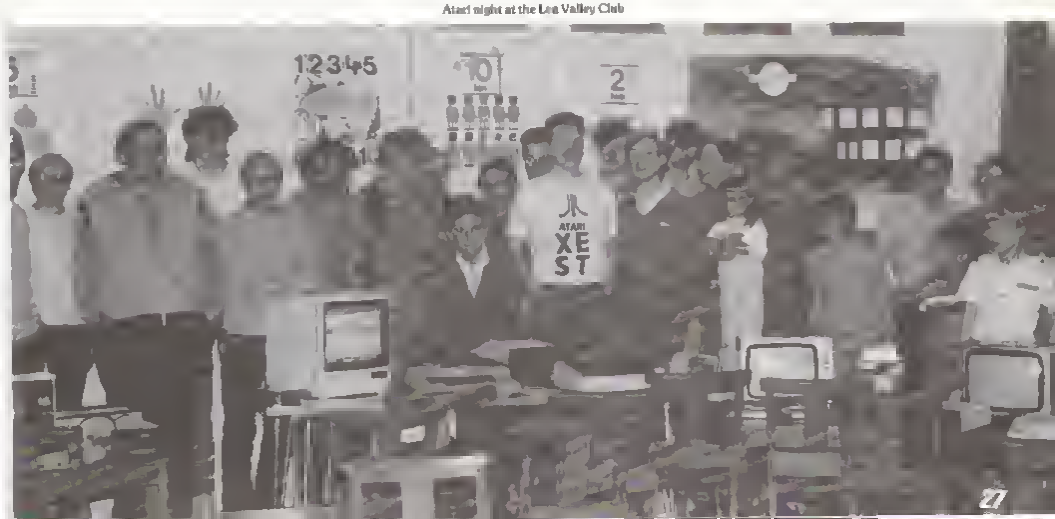
At the meetings, most members were keen to see whet software was around and so we had to have 3 or 4 machines constantly running with all the latest on display Unfortuneatly et this point, Nigel Fowler hed to leave us (pressure of work) and this meant a new President hed to be found. As often happens nobody wanted the job at the time but we managed to muddle through with yours truly et the helm for the next six months. Then one night we had a meeting in a local pub (I assure you that this was a one off, Hic!) and Ken Hewitt was voted in as our new President.

From here on it was all uphill, our newsletter got bigger end bigger (and better and better we hope) increasing from 4 to 20 pages. Our meetings became more regular and our ettendance grew by 50%, well almost! We started to give demonstrations of new software of all types, arcade utilities and educational. We gave help to people with programming problems in Basic and Machine Code, and we now include Logo (Pilot is not supported though but then its not that popular in the U K)

We cater for everyone including youngsters with no knowledge except how to play games and adults with no knowledge except how to play games (!) We have en age span from 10 to 60 but sadly only 2 lady members We have members ell over the country and indeed, all over the world, so enything is possible If you would like to join in the clubs activities and are within attendance distance (London to Hertford) why not call in on us? If you live elsewhere we can arrange a special membership deal (write for details)

Meetings are held at the Church Hall, Church Lane Wormley (near A10/M25) They are fortnightly commencing at 7 00 pm until 10 00 pm Please send a large SAE for details, or phone Waltham Cross 28168 (evenings only), we will be very glad to hear from you We would also like to hear from other nearby Atari Users Groups so that we could organise a Mega meeting how about it?

Atari night at the Lea Valley Club



WHAT'S MIDI?

by Michael Stringer Part 1

Looking through the specification of the Atari 520ST the other day I noticed that it read 512K RAM, Expandable ROM and then 'MIDI Interface' MIDI was placed third in the order of priority, way ahead of such exciting things as 'Hard disk Interface 16 Bit Motorola Microprocessor running at 8MHz', etc To be given such a prominent position it must be pretty important.

MIDI is an acronym for Musical Instrument Digital Interface which is a Communication Standard for musical instruments and is probably the most important single development devised for the electro musical fraternity It allows the hardware of this artform to communicate with itself and also the microprocessor, and in the case of the Atari 520ST an enormous amount of power is released.

The initial meetings between the major manufacturers began in 1981 and a draught proposal was put forward in April 1983 which gave rise to the MIDI DATA FORMAT The signatories of this document were Oberheim Sequential Circuits and Roland, who were also acting on behalf of Yamaha, Korg and Kawai These manufacturers did not want to fall into the same pit as computer manufacturers, they realised the importance of having a universal instrument interface Imagine how much more enjoyable it would be if all the manufacturers of say, 6502 based computers had produced a similar specification for a Basic Language! Anyway, they didn t and we haven't.

The proposal was thrown around a bit and MIDI specification 1 0 was published in August 1983 The first instrument to be manufactured with MIDI incorporated was the Prophet 600 and since then it can be found in Organs Electric Pianos, Polyphonic Synthesizers, Guitars Remote Keyboards Expansion Units Digital Sound Samplers, Digital Sound Sequencers, Tone Generators, Rhythm Units, Foot Pedals, Electric Drum Sets, the Yamaha CX5 MSX computer and the Atari 520ST Not bad, for something proposed less than two years ago, and this list is being added to almost daily Designers are constantly looking for more and more applications in which MIDI can be incorporated

Is it Limited?

The next question that needs to be answered is, "What can it do"? Before this question can be answered it must be appreciated that MIDI in its present format, does have its limitations It will not turn a cheap synthesizer into an all dancing singing, superduper top line (and very expensive) synthesizer, neither will it turn you into the greatest musician since Wolfgang Amadeus Mozart, but it will certainly open up a new, and very absorbing, world if you have any musical

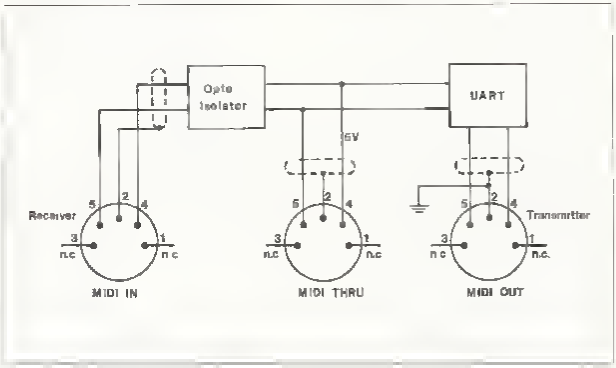


Figure 1

interest. The extent of its possibilities will be limited by the software availability If for example, you have a touch-sensitive synthesizer, there is no doubt that the Atari 520ST will be able to teach you to play correctly This is because the synthesizer will be able to communicate with the computer as well as the computer communicating to the synthesizer The computer will know exactly which note you have played in response to a given text and it will know exactly how you played the note. All the information regarding note, key velocity and pressure is sent out from the keyboard and can be analysed very critically

If learning to play does not appeal to you it is still possible to utilise the other applications such as linking a synthesizer to the computer and letting the computer play scores, or store real time information from the keyboard, If you can already play The computer can synchronise other synthesizers and ancillary apparatus such as rhythm units sequencers In fact anything from the list given earlier, up to a maximum of 16 The only proviso in this instance is that only one voice or channel is dedicated to each peripheral

The two essential features of MIDI are the Interface and the Control Data. In Figure 1 can be seen, in a simplified manner, the important features of the Interface As far as the user is concerned, it consists of three ports These are known as MIDI IN MIDI OUT and MIDI THRU (yes it is another American word we are stuck with!) These are standard 180 degree 5 pin female DIN sockets. The Atari 520ST only has two sockets, MIDI OUT and MIDI IN

The fact that there is only one pair of sockets is a very limiting feature but more about that at a later stage. There is no need

for a controlling device to have MIDI THRU which is why there are only two sockets to be found tucked at the back of the computer

MIDI DATA is Asynchronous, Serial, operating at 31 25 Kbaud, which is why MIDI data can be sent via DIN hardware, because it is obviously cheaper to manufacture than it would be if the data was sent parallel. The three modes of data transfer are MIDI IN, which receives data, MIDI OUT which transmits data and MIDI THRU provides a buffered output of data received from MIDI IN allowing data to be sent to the next piece of apparatus in a Train Network

The interface is a 5mA current loop, which is opto isolated The two states of the data HIGH (1) and LOW (0) produce current changes in the circuit When the data state is LOW current flows through the opto-isolator, effectively grounding the 5V normally sent to the UART causing the UART to see a logic LOW When the logic data goes HIGH the opto-isolator does not conduct and, therefore the UART sees logic HIGH There is no ground connection between a transmitter and receiver. The screened cable from the MIDI IN socket is not connected to the opto isolator (pin 2) It can be seen from the diagram that on all the sockets, pins 1 and 3 are not used The maximum length of cable that can be used is 15 M (49 Feet)

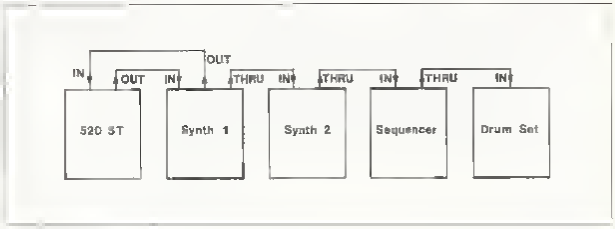


Figure 2

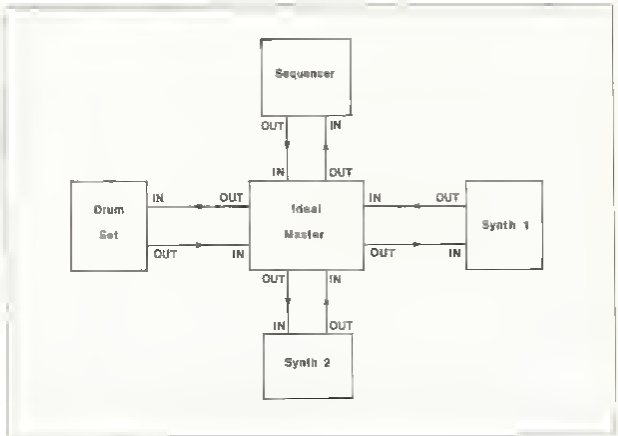


Figure 3

Systems

There are two ways of networking a system, these can be called the Train and the Wheel Typical examples are shown in Figures 2 and 3 Figure 2 is a possible arrangement using a Train Network. The 'Master' controller could be the Atari 520ST for example Two-way communication is only possible with one of the peripherals here it is connected to Synth 1. Data from the Master is transmitted to the MIDI IN of Synth 1 and the MIDI THRU from this connection is forwarded to the next peripheral The MIDI OUT of Synth 1 is connected to the MIDI IN of the Master It can be seen immediately from this arrangement there is a severe restriction enforced by having only one set of MIDI sockets There is a very cramped utilisation of the full potential of the Master MIDI is a two way communication system, and it can be seen that only one peripheral can utilise this facility

It is a great pity, that in their infinite wisdom, Atari could not have put some extra MIDI outputs on the 520ST to take full advantage of the tremendous power that is available This would enable a Wheel Network to be set up In Figure 3 a typical Wheel Network is shown This is without doubt the best arrangement to have all the peripherals can communicate with each other, utilising to the full, the enormous potential that MIDI can provide If Atari had increased the number of outlets it would have turned a great computer into an outstanding one!

Next time we will look at the Data Format and the structure of the MIDI Language.

Bootbase

Continued from page 26

```
YY 750 POKE 764,255:? "ENTER DISK # (4 CH
   RRS)"
GG 755 INPUT D$:IF D$="" THEN 755
TU 756 ? "ENTER SIDE (A,B)"
KS 757 INPUT S$:IF S$="A" OR S$="B" THEN
   760
UL 758 GOTO 757
NT 760 IF LEN(D$)=4 THEN 780
KI 770 D$(LEN(D$)+1,4)="   "
ZT 780 RETURN
NR 800 ? :? "THIS IS NOT A VALID MULTI800
   F DISK":NRIT=2500:GOSUB 1100:POP :GOTO
    205
NP 900 POKE 764,255
JN 910 IF PEEK(764)<>12 THEN 910
ZJ 920 RETURN
LE 1000 ERR=ERR+1:IF ERR<5 THEN 340
NG 1010 ? :? "Error in reading your Multi
   boot disk":? "Please check it and re-t
   ry":NRIT=2500
ZP 1020 GOSUB 1100:GOTO 205
XK 1100 FOR I=1 TO NRIT:NEXT I:RETURN
FV 1200 POKE 764,255
IG 1210 IF PEEK(764)=7 THEN 600
UR 1220 IF PEEK(764)<>12 THEN 1210
RM 1230 RETURN
XG 1300 ? :? "THANK YOU FOR USING BOOTBAS
   E"
OR 1310 ? :? "BYE FOR NOW":POKE 580,1:CLO
   SE #1:OPEN #1,4,0,"K:":? :? :? :? :? :
   GOTO 3000
GU 1600 ? :? "500 ENTRIES IS THE MAXIMUM"
CV 1610 ? "FOR SAVING TO D:PROGLIB.DB"
PE 1620 ? :? "LIMIT NOW REACHED !"
NL 1625 ? "(You have a lot of programs ..
   .)"
NJ 1630 NRIT=2500:GOSUB 1100:GOTO 205
UX 2000 ? :? "ERROR IN PRINTING":? :GOTO
   360
DR 2500 ERR=ERR+1:TRAP 2500:IF ERR>1 THEN
   2510
OX 2505 CLOSE #1:OPEN #1,8,0,"D:PROGLIB.
   DB":GOTO 452
CI 2510 ? :? "ERROR IN ACCESSING DOS 2.0
   ORTR DISK":? :GOTO 605
DE 3000 ? :? "PRESS RETURN TO RE-BOOT":PO
   KE 752,1:GOSUB 900:X=USR(58487)
```

THE HAPPY TYPER

by Steve Hillen

Introduction

The Happy Typer is a utility for use with Atari Basic. It will give you automatic intelligent line-numbering and 10 extra keys which you can redefine to print out keywords thus speeding up your typing. Unlike many auto line numbering facilities this one allows the full use of the Atari screen editor, so you can adjust lines while still in the auto mode. The redefined keys are accessed by pressing the SHIFT and CONTROL keys simultaneously with a number key. These keys are not used by Basic or the operating system, so you can still type in all those control characters safely.

Making a Copy of Happy Typer

If you are using a cassette only system, then type in listing 1 using Basic. Save out this program before doing anything else. Type RUN and the program will check your typing and ensure that the data is correct. Retype those lines that produce an error. Once all is correct the program will prompt you to ready a new cassette. On typing RETURN the program will save out an autoboot file which is the Happy Typer. Load this file by pressing START on power-up with Basic. Happy Typer will load, tell you that it's OK, and be ready for use.

If you are using a disk system, type in listing 2 and save it out. RUN it and correct any mistakes found by the program. Once it's ready the program will ask you to insert a disk with DOS on it. The program will then save out an Autorun.Sys file onto the disk. Don't change the filename — Happy Typer will only work as an Autorun.Sys file. The next time you boot this disk with Basic, Happy Typer will be ready for use.

Using Happy Typer

1-The Auto Line Numbering

Every time you tap the TAB key after a RETURN, a new line number will be printed. If you type on the TAB key and the last key pressed was not a RETURN then the TAB will be performed as normal. This is better illustrated by example.

Directly after power-up, press the TAB key — the first line number will be printed. Type ?"hello" then RETURN then TAB. The next line number will then appear. Play around and get used to using the TAB key after a RETURN. If you type in a few lines of Basic, then list them, then press RETURN and TAB the next line number after the last line of your program is printed. Also, if you use the cursor keys to move to another line and press RETURN on that, then the next auto line number will be that plus 10.

This method of auto line numbering remembers the last line you typed, and gives you the next one if selected with the TAB key. If you want to type in a new number, just do so without using the TAB key.

Finally, to change the increment of the auto line numbering just type
INC nn where nn is any number you like. e.g. INC 2000 will give line numbers 2000 apart. INC 1 will give numbers one apart.

2-The Redefined Keys

The keys that can be redefined are the row of 10 numbers across the top of the keyboard. To redefine a key type
DEF 1 ?"hello"

Every time you type SHIFT CONTROL 1 simultaneously, ?"hello" will be printed. Another example: DEF 8 POKE. Now a shift-control 8 will type out POKE for you. Get the idea?

Note that the space between DEF,1 and the string are important. Also note that each key is allocated only 16 characters, so this is the maximum you can stuff onto one key. If you redefine a key that has already been redefined, then the last redefinition will be printed. If you should wish to delete a key just type DEF 1 then RETURN without the second space. You will find that all characters except trailing spaces and the return key can be printed, so you might set up one key to backspace say 10 characters by using the Escape Cntl cursor keys.

Finally, don't worry about hitting the System Reset key, the program is safely installed and protected, and will remember all the keys you've defined, and the increment and current line number.

How does it work?

The program falls into two sections. Firstly there is an editor patch. The editor is located in the device table, and its vectors are moved into RAM. I adjust the get-byte vector to point to my new routine. This new routine waits for a return to be typed, then scans the input buffer for either INC or DEF. If neither are found then the line is passed back to Basic as a normal line. If one is found, then the operation is performed, and the line is not passed to Basic. The second section is a patch into the keypress routine. The keypress interrupt vector is stolen and the new routine looks for a shift-control number or a TAB immediately following a return. If a defined key is detected, then the string is printed out a byte at a time through the editor put byte routine. If a TAB is found, then the input buffer is examined for a line number and the increment is added to form the next number which is then printed via the put-byte routine.

The program is just over 3 pages long and protects itself by altering MEMLO. Don't change MEMLO or poke anywhere beneath it!

Note that this is only an editing aid and is irrelevant when running the program. The cassette version loads over pages 7-10, and the disk version loads over pages 31-34, so page 6 is left free for your own use.

Happy Typing!

Listing 1

```
EI 1 REM ************************************
NH 2 REM
QZ 3 REM HAPPY TYPER (CASSETTE VERSION)
NJ 4 REM
TF 5 REM BY STEVE HILLEN
NL 6 REM
ZT 7 REM MONITOR MAGAZINE 1985
NN 8 REM
EQ 9 REM ************************************
DW 10 DATA 0,1,2,3,4,5,6,7,8,9,0,0,0,0,0,0,
   0,0,10,11,12,13,14,15
MM 15 DIM DAT$(91),HEX(22)
KZ 20 FOR X=0 TO 22:READ D:HEX(X)=D:NEXT
   X:LINE=990:NESTONE 1000:TRAP 60:? "Che
   cking data":PRINT
LI 25 LINE=LINE+10:? CHR$(26);"Line=";LIN
   E:READ DAT$:IF LEN(DAT$)<>90 THEN 110
VQ 28 DATLINE=PEEK(183)+PEEK(184)*256:IF
   DATLINE<>LINE THEN ? "Line ";LINE;" m
   issing.":END
LU 30 FOR X=1 TO 89 STEP 2:D1=ASC(DAT$(X,
   X))-48:D2=ASC(DAT$(X+1,X+1))-48:BYTE=H
   EX(D1)*16+HEX(D2)
JB 35 IF PASS=2 THEN PUT #1,BYTE:NEXT X:H
   EAD CHKSUM:GOTO 25
BL 40 TOTAL=TOTAL+D1+D2+96:IF TOTAL>999 T
   HEN TOTAL=TOTAL-1000
CE 45 NEXT X:READ CHKSUM:IF TOTAL=CHKSUM
   THEN 25
NE 50 GOTO 110
PE 60 IF PEEK(195)<>6 THEN 110
IT 65 IF PASS=2 THEN CLOSE #1:? "Done it.
   ":END
UE 70 ? "Ready cassette and press <return
   >":OPEN #1,8,128,"C:"
```

```
HH 90 PUT #1,255:PUT #1,255:PUT #1,0:PUT
   #1,31:PUT #1,176:PUT #1,3A
CN 100 ? "Writing file"; ? :PASS=2:LINE
   =990:RESTORE 1000:TRAP 60:GOTO 25
AZ 110 ? "Bad data on line";LIAE:LIST LI
   AE:? TOTAL:END
PE 1000 DATA 0DD080B6D007A93CBD0D3A9BCBD
   150AA9D08D4402A200A9D99D4203A906D4503
   A9059D4403A9089D4903A97F,990
DA 1010 DATA 9D46D32056E4186048617070729
   54797B6572204F4B2E9
   ,610
BK 1020 DATA
   0603
   DDDDDD29DDD8A9468DE702A9,90
HQ 1030 DATA BABDE80A2A200D1A03E8E0E0C945
   D0F6BD1E0D3B5CB8D19D3B5CCA91D9D1E0A9BA
   9D19D3ADBFD1C9911E0ADD1D,255
KS 1040 DATA F8AD1C0A1B6901E0D4DFBAD1D0A69
   B0D04E08A9089D1DBAA94BD1CDAAD1E0A1B69
   01E0DA89AD1FDA69006D80F,430
AK 1050 DATA 78AD0802BDA207AD0902BDA307A9
   07BDD9D2A973BDD8D2581B4070D8E04D9E40AC
   150AADD9D29D15DAA2D9DD41,519
```

```
IJ 1060 DATA 0AFD13CA10F9C92CD004CD0CF029
   60A068AA58ACFFFFD61009F018DD6009A0A0A
   AADAAA8E5F09D16D9202D9D9,733
NQ 1070 DATA AE5F09EBCE5D09DEE4C9C07A2FF
   EBDDD80AC930F0F06E160A2009D9EE160AAE16
   DABDD80AE00598F0A9210DFC,8
HL 1080 DATA D26BAB6BAA6B5B40A2FFE0305E0D
   B005C92F0F4C93DF0F2D94C93ABD45EDADDD
   99080A8DDDD5297FC93D9DC,109
ZH 1090 DATA C93AE009E0CBC0059BEAF014A204
   09060A9D0B0ACAE01DF6A93D9D0DDACA1DFA1B
   A2D4BD1DDA7DDBDAC93A9DD3,265
AB 10D0 DATA E9BA3B9DDDDACA1BED6D20FFFFDB
   C998D8308A4B984870F0075BA2008E400AA0
   0009B005DD2BDAD008E8C8C0,396
YY 10E0 DATA D39DF2BD1DC8EBC803D0FAEE400A
   EDD59DE1A9FF6D170A68A868AAA99BEE170A28
   600C17DAA99B999DB5DD8005,580
AH 10F0 DATA AE4DBABD3DDABDDD30BBD2E0A8DB4
   B82DB20BA2DBBE17DA7D1AA7999D4203BE4903
   A97F9D4803A90A9D4503A932,676
BY 1100 DATA 9D448D2056E44C4DAD4CFFFFADFF
   C8B9804B5C99BFDB0C93B9004C93A9DF03BDD1E
   B03DFAA2D4B9D4D529DF9D1D,848
```

```
CH 1110 DATA DACA9010F4E0083D80D9B09D100A
   CA10FA1B6020FFFFE0A0049B0D5C93D9046C9
   3ABD4238E93D480A0A0A0AAA,809
LU 1120 DATA A9BBD5DE9CBB9B085C998D00960
   AAA9B09D6109F021C889BD5C998FD119D6D09
   EBEE5ED9AD5ED9C91090EACE,93
WC 1130 DATA 5ED960AAAD5ED99D61D916B03B60
   BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
   BBBBBBBBBBBBBBBBBBBBBBBB,647
DK 1140 DATA BBBBBBBBBBBBBBBBBBBBBBBBBBBB
   BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
   BBBBBBBBBBBBBBBBBBBBBBBB,967
JZ 1150 DATA BBBBBBBBBBBBBBBBBBBBBBBBBBBB
   BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
   BBBBBBBBBBBBBBBBBBBBBBBB,287
UC 1160 DATA BBBBBBBBBBBBBBBBBBBBBBBBBBBB
   BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
   BBBBBBBBBBBBBBBBBBBBBBBB,607
DM 1170 DATA 0000000000000030303031300BBB
   0001000000000000000000000000000000000
   BBBBBB494E4344454680BB9D5,50
CU 1180 DATA 0D53796E74617B2D6572724F722E
   9B00F2DFDEDADBDD0F3F5FD0000000000000
   0000000000000000000000000,924
```

Listing 2

```
EI 1 AEM ********************************
NH 2 AEM
AE 3 REA HAPPY TYPER (DISK VERSION)
NJ 4 AEM
TF 5 AEA BY STEVE HILLEA
AL 6 AEM
ZT 7 AEA ADNITDR MAGAZIAE 1985
NA 8 AEM
EG 9 AEA ********************************
DA 10 DATA 0,1,2,3,4,5,6,7,8,9,0,0,0,0,0,0,
   0,0,10,11,12,13,14,15
AN 15 DIM DAT$(91),HEX(22)
XZ 20 FOR X=0 TO 22:READ D:HEX(X)=D:NEXT
   X:LINE=990:RESTORE 1000:TRAP 60:? "Che
   cking data":PAINT
LI 25 LIAE=LIME+10:? CHR$(29);"Line:";LIN
   E:AEAD DAT$:IF LEA(DAT$)<>90 THEA 110
VD 20 DATLINE=PEEK(183)+PEEK(184)*256:IF
   DATLINE<>LIAE THEA ? "Line ";LIAE;" m
   issing.":END
LU 30 FOR X=1 TO 89 STEP 2:D1=ASC(DAT$(X,
   X))-48:D2=ASC(DAT$(X+1,X+1))-48:BYTE=H
   EX(D1)*16+HEX(D2)
JW 35 IF PASS=2 THEN PUT #1,BYTE:NEXT X:R
   EAD CHKSUN:GOTO 25
BL 40 TOTAL=TOTAL+D1+D2+9:IF TOTAL>999 T
   HEN TOTAL=TOTAL-1000
CE 45 AEXT X:READ CNKSUN:IF TOTAL=CHKSUN
   THEN 25
ME 50 GOTO 110
PE 60 IF PEEK(195)<>6 THEN 110
IL 65 IF PASS=2 THEN PUT #1,224:PUT #1,2:
   PUT #1,225:PUT #1,2:PUT #1,76:PUT #1,3
   4:CLOSE #1:? "Done it.":END
```

```
HA 70 ? "Insert disk with DOS.Press <retu
   rn>.";:DIA IA$(1):INPUT IN$:OPEA #1,8,
   0,"D:AUTOAUA.SYS"
HH 90 PUT #1,255:PUT #1,255:PUT #1,0:PUT
   #1,31:PUT #1,176:PUT #1,34
CN 100 ? "Writing file"; ? :PASS=2:LIAE
   =990:AESTOME 1000:TRAP 60:GOTO 25
AZ 110 ? "Bad data on line";LINE:LIST LI
   AE:? TOTAL:EAD
FC 1000 DATA 20FFFFDBA94EBDE7D2A922BDEBD2
   A2D0BD1AD3E8E8E0C945D0F6BD1BD3D5CBBD19
   D3B5CCA919D1B03A9229D19,243
AV 1010 DATA 83ADDF01C0991B22B01DFBAD1F22
   1B6901BD5B2BAD2B2269B0D5120A920BD2B22
   A94E8D1F22AD21221069B1BD,232
XY 1020 DATA BD21AD22226990B5DBE2179ADD902
   8DA51FADD9D2BDA61FA91FBDD902A97E0DDBD2
   5B1B6B07BD9BA4B954BAC1B22,310
AK 1030 DATA AD09D20D1022A209DD4422FD13CA
   10FBC92CDBB4CB0CFD2F6A8A868AA58A4CFFFFBD
   6421FD1B3D6321BABABABADA,581
XP 1040 DATA AA8E6221B0D6E212B0C21AE6221EB
   CE6321D0EE4C9F1FA2FFE90D0E22C930FDFBBE
   19222B0C21EE1922AE1922BD,707
EP 1050 DATA 0E22E00590FDA921BDFCD26BAB6B
   AA6B5040A2FFE0305A0D0005C920F0F6C930F0
   F29049C93ABD45EBADDD99DE,823
XK 1060 DATA 220D0DD5297FC93D9DCC93ADD0B
   E0C8C0059DEAFD14A204890E229D0E22CA8010
   F6A993D9DE22CA1BFA1BA2D4,926
CD 1070 DATA 6D13227D0E22C93A9D03E98A309D
   BE22CA1DED6D20FFFF08C999D0308A4B984870
   2DF31F50A2D00E4322ADDDE9,31
BS 1080 DATA BB05DD2022D009E8C0C009F2B0
   1DCBEBCDD3DDFAEE4322ED059DE1A9FFBD1A22
   69A868AAA999EE1A222060AC,229
```

```
RD 1090 DATA 1A22A99899B0858D8005AE4322BD
   33220D0620BD31220DD7202BD528A200E1A22
   9D1AA9B99D4203BE49B3A97F,251
GK 1100 DATA 9D4803A9229D4503A9359D440320
   56E44C4DAD4CFFFFADFFCB09B405C998F009C9
   309804C93A90F03B8D1E0830,393
AB 1110 DATA FAA2D4B9B485290F9D1322CAEB1D
   F4EDDD3DDBA9DD9D1322CA1DFA1B6B2DFFFF6D
   ADD4B9BD5C93D9D46C93ADD,427
IB 1120 DATA 423BE93B480ADADABAAAA9DBBD61
   21CBE9BD5C998D0D9A6AAA9DB9D6421FD21C0
   B9BDD5C99DFD119D6E21EBEE,543
DU 1130 DATA 6121AD6121C9:09BEACE612160AA
   AD61219D6421186035B4B0D0D0D0D0D0D0D0D0
   0000000000000000000000,215
TA 1140 DATA BBBBBBBBBBBBBBBBBBBBBBBBBBBB
   0000000000000000000000000000000000000
   BBBBBBBBBBBBBBBBBBBBBBBB,535
NH 1150 DATA 0000000000000000000000000000
   BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
   0000000000000000000000,855
TW 1160 DATA 0000000000000000000000000000
   BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
   0000000000000000000000,175
SD 1170 DATA 0000000000000000000000000000
   0000003B3D3D313B0000000100000000000000
   BBBBBBBBBBBBBBBBBBBBBBBB,512
AF 1180 DATA BB494E4344546202D1D81B53796E
   74617B2D6572726F722E9B00F2DFDEDAD0DBB
   F3F5F0A90C0D1822A50C8D01,620
JT 1190 DATA 1FA5B09D021FA990B50CA91F850D
   A900D6448A2D0A909D42D3A9229D4503A988
   9D440A3A900D9A903A97F9D40,684
HR 1200 DATA 032856E44C031F4361707072054
   797B6572204F4B2E9B000000000000000000
   0000000000000000000000,297
```

BACK ISSUES

Previous issues of this magazine are obtainable from the club for £1 plus 30p postage each. They contain many interesting and informative articles, hints & tips, program listings for you to input, reviews and practical advice. If you have missed out send for your copies of back issues today! Please note that issues 1,2,3 & 7 are already sold out.

Issue 4.

Includes a complete in-depth look at Display Lists, what they are, how to use them, LMS explained, horizontal and vertical scrolling, etc. Another article shows how to get text on a Graphics 8 screen and gives an example graph to prove the point. A comprehensive review of many of the different types of joystick that are available gives ratings for comfort, action, looks and value. Program listings are aplenty and include 'Peckman' a Basic version of a well known arcade game, Stunt Rider in which you must jump your motorbike over the buses. Hex is a two player board game with excellent graphics, and for the more serious minded you can even enjoy designing your own shapes with CAD (computer assisted design).

Issue 5.

The first part of the series on 'Cracking the Code' starts in this issue and covers Binary, Hexadecimal and Decimal mathematics. There is an article on protecting your Basic programs from prying eyes and an interesting article on hardware modifications to the 800/400 machines to give improved sound and picture quality, a cold start key and a busy light for your cassette player. Also included is a review of the new programming language 'Action!' showing its potential for creating exciting fast action games. Games listings shown include Gil-bert, which is a 'Q-bert' type game, also Dragonfire in which the player must cross the drawbridge dodging the dragons flaming breath to reach the treasure room. Other listings include a label maker and a QRA locator for Radio Amateurs.

Issue 6.

Includes a useful tutorial showing how to print Micropainter and Versawriter pictures, also contains a terrific program demonstrating 80 characters across the screen. A new regular column for adventure enthusiasts is started to give reviews of adventure games and give hints and tips on how to play them. Part two of Cracking the Code continues with addressing modes and binary sums. The hardware design for a Light Pen is shown together with some simple programs to use with it once you have built it. Fun with Art from Epyx is reviewed and some of the excellent results of using this package are shown. Programs include Planetron and a RTTY listing for use with a short wave band radio, the Atari 850 interface and a signal terminal unit (such as the Maplin TU1000).

Issue 8.

Contains a preview of the new Atari computers. Two new series start, one about how files work and the other 'Starting from Basics' for beginners. Cracking the code continues and the concluding part of Interrupts discusses horizontal and vertical scrolling. The adventure column includes reviews of Mask of the Sun and Sorcerer. Other reviews include Conan, Spy vs Spy, Alley Cat and Ghostbusters. Programaaare Matchbox, a concentration game, Quick-plot a Graphics 8 Plot/Draw-to utility and Nightmare Reflections, an exceedingly frustrating adventure.

Cracking the Code
Continued from page 21

equal (BEQ) instruction will take us to EXIT which returns to BASIC. The test is made first in case the multiplier is zero, if so then the result will also be left as zero. Assuming the multiplier isn't zero then lines 320 to 350 clear the carry and add the multiplicand to the low byte of the result. If the carry is left clear then a branch is made to NOCARY, else one is added to the high byte of the result by the increment instruction on line 370. This method of seeing if the carry is set and then incrementing the high byte by one is exactly the same as adding with carry (ADC) zero to the high byte; however, this would require a load and a store instruction, thus we save two bytes! All that happens then is one is taken from the count by the decrement X instruction and a jump is made back to the comparison instruction at MLTPLY to continue the multiplication. The loop will continue like this until the count in the X register has been taken to zero, once this happens it will return to BASIC and the answer, held at RESULT and RESULT+1, will be placed into BASIC's variable.

Listing 7

```
10 DIM HEX$(16)
20 LINE=10000:TRAP 100
30 READ HEX$,CHKSUM:SUM=0:J=0:START=
1536
40 FOR I=1 TO 15 STEP 2
50 D1=ASC(HEX$(I,I))-48:D2=ASC(HEX$(
I+1,I+1))-48
60 NUM=((D1-7*(D1>16))*16+(D2-7*(D2>
16)))
70 SUM=SUM+NUM:POKE START+J,NUM:J=J+
1:NEXT I
80 IF SUM=CHKSUM THEN LINE=LINE+10:G
OTO 30
90 PRINT "Checksum error on this lin
e:"
95 LIST LINE:END
100 PRINT "Data in memory."
10000 DATA 68686885CB686BAA,1026
10010 DATA C5CB9005A5CB86CB,1254
10020 DATA AAA900B5D485D5E0,1254
10030 DATA 00F00F18A5D465CB,960
10040 DATA B5D49002E6D5CA4C,1212
10050 DATA 17866000000000,125
```

Running the Program

To run the program it will have to be in the locations starting at 600 hex (page 6). If you assembled your own copy of this program then you could load the object file into memory for disk owners life is simple just nip into DOS and binary load the object file, then jump back to BASIC. If you are using the assembler editor cartridge with cassette then beware of an error. You cannot CLOAD the object file as is stated in the manual instead you will need a short routine (see the one on page 8 of issue 6) such utility routines will be delt with in the next part of this series.

If you can't load your object program or don't have an assembler yet, then Listing 7 is a BASIC program which reads the hex data and stores them in memory after they have been converted to decimal for the POKE statement, if you have made a mistake in the data a 'checksum error' will be printed out if so, then re-check the DATA statements. Having saved the program, type RUN and the machine code in the data will be poked into memory, after a short delay the message 'Data in memory' will be displayed. You can now test it by typing

ANSWER=USR(1536,10,24)

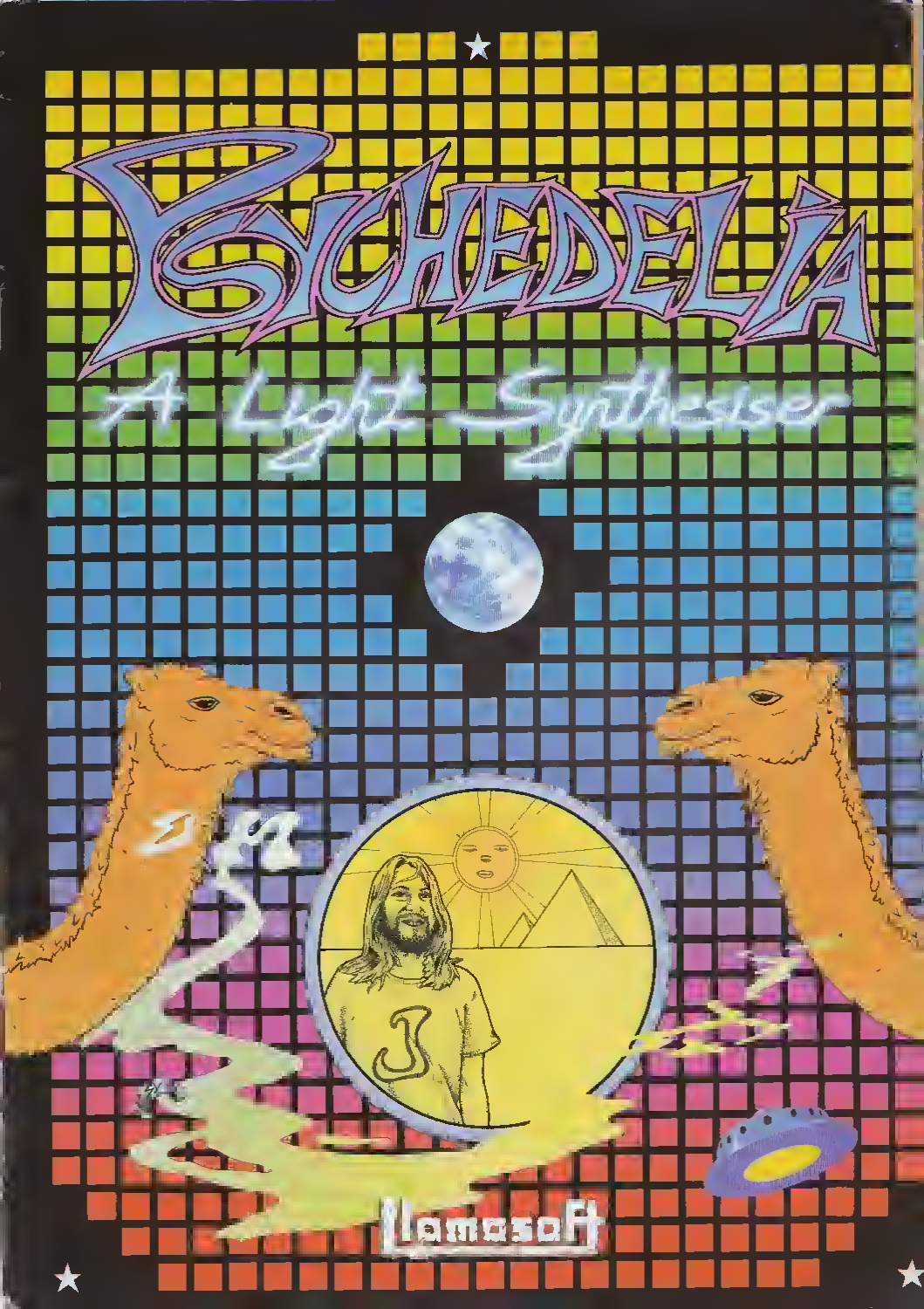
then typing ?ANSWER will give the result of 240. You can change the two parameters, but don't alter the first number of 1536, this is the decimal for 600 hex, which is the start of the program.

Until Next Time...

In the next issue I will tie up some loose ends, including looking at an improved multiplication routine, and then start some new programs and topics. Have fun experimenting until then, and if you haven't already done so, rush out and buy that assembler you were promising yourself!

PSYCHEDELIA

A Light Synthesiser



ATARI ST

520ST

POWER WITHOUT THE PRICE

THE NEW ATARI 520ST

USER FRIENDLY GEM OPERATING SYSTEM

FREE SOFTWARE AND FUTURE EXPANSION

Silica Shop Price: £651.30 + £97.70 VAT = £749.00. This price includes:

- ★ 512K RAM
- ★ MOUSE
- ★ GEM
- ★ B/W MONITOR
- ★ 500K 3.5" DISK DRIVE
- ★ KEYBOARD (95 KEYS)

£749

ATARI 520ST SPECIFICATION

MEMORY

GRAPHICS

ARCHITECTURE

DATA STORAGE

DISK DRIVE

SOUND AND MUSIC

KEYBOARD

MONITOR

VIDEO PORTS

COMMUNICATIONS

GEM WIMP ENVIRONMENT

SOFTWARE

VARIOUS

LANGUAGES

MACINTOSH v F1e v 520ST

	APPLE MACINTOSH	APRICOT F1e	ATARI 520ST
PRICE	£2,984	£1,382	£749

PRESS COMMENT

SILICA SHOP

WE ARE THE UK's No1 ATARI SPECIALISTS

- FREE POST & PACKING ON MAIL ORDERS
- FREE NEXT DAY SECURICOR DELIVERY
- INFORMATION MAILING SERVICE
- TECHNICAL SUPPORT TEAM
- HIGHLY COMPETITIVE PRICES
- AFTER SALES SUPPORT SERVICE
- REPAIR SERVICE ON ATARI PRODUCTS

If you would like to be registered on our mailing list as an Atari computer owner, or as a person interested in buying an Atari machine, let us know. We will be pleased to keep you up to date with new Atari developments free of charge. So, return the coupon today and begin experiencing a special Atari service that is second to none.

SILICA HOTLINE ☎ 01-309 1111

SILICA SHOP LTD, 1-4 The Mews, Hatherley Road, Sidcup, Kent, DA14 4DX

SEND FOR FREE ATARI ST LITERATURE

To: Silica Shop Ltd, Dept ATCOC 0685, 1-4 The Mews, Hatherley Road, Sidcup, Kent, DA14 4DX

PLEASE SEND ME FREE LITERATURE

ON THE NEW ATARI 520ST COMPUTER

Mr/Mrs/Ms: Initials: Surname:

Address:

Postcode:

Do you already own a computer
If so, which one do you own?